IEEE Std 730.1-1995
(Revision and redesignation of
IEEE Std 983-1986)

IEEE Guide for Software Quality Assurance Planning

Sponsor

Software Engineering Standards Committee
of the
IEEE Computer Society

Approved December 12, 1995

IEEE Standards Board

Abstract: Approaches to good Software Quality Assurance practices in support of IEEE Std 730-1989, IEEE Standard for Software Quality Assurance Plans, are identified. These practices are directed toward the development and maintenance of critical software, that is, where failure could impair safety or cause large financial losses.
Keywords: software life cycle processes, software metrics, software quality assurance plan

---

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 1-55937-593-0

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. 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.

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.

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 all 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 technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

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

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

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

Introduction

(This introduction is not a part of IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning.)

The purpose of this guide is to identify approaches to good Software Quality Assurance practices in support of IEEE Std 730-1989, IEEE Standard for Software Quality Assurance Plans. This guide is meant to supplement IEEE Std 730.1-1989 by presenting the current consensus of those in the software development community who have expertise or experience in generating, implementing, evaluating, and modifying Software Quality Assurance Plans. This guide is not offered as a detailed procedures manual for establishing and operating Software Quality Assurance programs. This guide does not constitute further requirements than those stated in IEEE Std 730-1989. An organization can claim compliance with IEEE Std 730-1989 without following this guide completely, or in part. Detailed information regarding specific software quality assurance activities may be found in the other IEEE standards. These are referenced where appropriate. While this guide quotes major portions of IEEE Std 730-1989, the standard is not quoted in its entirety. IEEE Std 730-1989 users are advised to consult that standard directly.

In accordance with IEEE Std 730-1989, the practices herein are directed toward the development and maintenance of critical software, that is, where failure could impair safety or cause large financial losses. Determination of this criticality lies in the "eyes of the beholder." The specific application and situation of each user must be carefully considered. Should there be doubt, it is suggested that the software be considered critical. For software that is definitely noncritical, or software already developed, a subset of the requirements stated in IEEE Std 730-1989 is appropriate.

This guide serves the three groups discussed in the Foreword to IEEE Std 730.1-1989: the user, the developer, and the public.

a) The user, who may be another element of the same organization developing the software, has a need for the product. Further, the user needs the product to meet the requirements identified in the specification. The user thus cannot afford to assume a "hands-off" attitude toward the developer and rely solely on a test to be executed at the end of the software development time period. If the product should fail, not only does the same need still exist, but also a portion of the development time has been lost. Therefore, the user needs to obtain a reasonable degree of confidence that the product is in the process of acquiring required attributes during software development.
b) The developer needs an established standard against which to plan and to be measured. It is unreasonable to expect a complete reorientation from project to project. Not only is it not cost-effective, but, unless there exists a stable framework on which to base changes, improvements cannot be made.
c) The public may be affected by the users' use of the product. These users include, for example, depositors at a bank or passengers using a reservation system. Users have requirements, such as legal rights, which preclude haphazard development of software. At some later date, the user and the developer may be required to show that they acted in a reasonable and prudent professional manner to ensure that required software attributes were acquired.

This guide is addressed to readers who have professional experience in quality assurance, or in software development, but not necessarily in both. For example, this guide should be useful to the following individuals:

a) A quality assurance person with the responsibility for developing or implementing Software Quality Assurance Plan for a project.
b) A software development project manager desiring to initiate Software Quality Assurance procedures on a project.
c) A purchaser or user of a software product who wants to evaluate a seller's Software Quality Assurance Plan or to specify a Software Quality Assurance Plan.
d) An independent evaluator, such as an electronic data processing auditor.
e) The person with accountability for implementation of a Software Quality Assurance Plan.

In the body of this guide, the use of “shall” is to be understood as referring to an item or activity that is mandated by IEEE Std 730-1989. The use of “should” is to be understood as referring to an item or activity for which there is a professional consensus. The use of “may” is to be understood as referring to an item or activity that can be advised under some circumstances, but for which there is not a professional consensus. The use of “could” is to be understood as suggesting the existence of several possibilities, the selection among which will be specific to the project and not driven by specific quality assurance considerations.

At the time this standard was completed, the Working Group on Software Quality Assurance Planning had the following membership:

Camille S. White-Partain, *Chair*
Roger G. Fordham, *First Vice Chair*
Robert B. Kosinski, *Second Vice Chair*
John P. Franklin, *Third Vice Chair*
Pamela Hinds, *Fourth Vice Chair*
Manfred Hein, *Fifth Vice Chair*

Terry L. King provided administrative support. Other individuals who have contributed review and comments include the following:

Fletcher J. Buckley
John W. Horch
Leonard L. Tripp
Christer Von Schantz
R. Jerrold White-Partain

The following persons were on the balloting committee:

Mark Amaya
Wolf Arfvidson
Theodore K. Atchinson
David Avery
Motoei Azuma
Bakul Banerjee
Richard L. Barrett
Mordechai Ben-Menachem
H. Ronald Berlack
B. P. Bhat
Robert V. Binder
Michael A. Blackledge
William J. Boll
Sandro Bologna
Damien P. Brignell
Fletcher Buckley
David W. Burnett
W. Larry Campbell
John P. Chihorek
Tsun S. Chow
François Coallier
Peter G. Comer
Christopher Cooke
Geoff Cozens
Patricia W. Daggett
Michael A. Daniels
Taz Daughtrey
Paul I. Davis
Bostjan K. Derganc
Rodney Dorville
C. Einar Dragstedt
Julian Edelman
Leo G. Egan
Caroline L. Evans
Richard L. Evans
John W. Fendrich
Judy Fiala
A. M. Foley
Gordon Force
Roger G. Fordham
Jay Forster
Kirby Fortenberry
Richard C. Fries
Yair Gershkovitch
Julio Gonzalez Sanz
Praveen Gupta
David A. Gustafson
Carol J. Harkness
Robert T. Harley
Manfred Hein
Mark Heinrich
Daniel E. Hocking
John W. Horch
Jerry Huller
Peter L. Hung
Lynn D. Ihlenfeldt
Richard Johansson
Ken Johnson
Russell E. Jones
Frank V. Jorgensen
Michael Kalecki
Laurel V. Kaleda
Myron S. Karasik
Ron S. Kenett
Judy Kerner
Peter Klopfenstein
Dwayne L. Knirk
Roy W. Ko
Shai Koenig
Robert Kosinski
Joan Kundig
Thomas M. Kurihara
Lak Ming Lam
Robert A. C. Lane
John B. Lane
J. Dennis Lawrence
Mary Leatherman
Randal Leavitt
Suzanne Leif
Fang Ching Lim
Bertil Lindberg
John Lindgren
Ben Livson
Dieter Look
John Lord
Joseph Maayan
Harold Mains
Ivano Mazza
Mike McAndrew
Patrick McCray
Russell McDowell
Jerome W. Mersky
Dennis E. Nickle
Michael O’Neill
Robert Parys
Paul G. Petersen
Donald J. Pfeiffer
John G. Phippen
Peter T. Poon

Thad L. D. Regulinski
James R. Roberts
Patricia Rodriguez
R. Waldo Roth
Sergiu Samuel
Benson H. Scheff
Norman Schneidewind
Wolf A. Schnoege
David J. Schultz
Leonard W. Seagren
Carl S. Seddio
M. C. Shaw
Alfred R. Sorkowitz
Vijaya Krishna Srivastava
Richard Staunton
Richard H. Thayer
Douglas H. Thiele
Leonard L. Tripp
Mark-Rene Uchida
Margaret C. Updike
Tom Vaiskunas
Glenn D. Venables
Spyros Villios
Udo Voges
Christer Von Schantz
Dolores Wallace
William M. Walsh
John W. Walz
Jack Ward
Richard Werling
Marleen White
Camille S. White-Partain
Scott A. Whitmire
Charles Wilson
Paul A. T. Wolfgang
Natalie C. Yopconka
Weider D. Yu
Janusz Zalewski
Geraldine R. Zimmerman

When the IEEE Standards Board approved this standard on December 12, 1995, it had the following membership:

E. G. "Al" Kiener, *Chair* **Donald C. Loughry,** *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril
Clyde R. Camp
Joseph A. Cannatelli
Stephen L. Diamond
Harold E. Epstein
Donald C. Fleckenstein
Jay Forster*
Donald N. Heirman
Richard J. Holleman
Jim Isaak
Ben C. Johnson
Sonny Kasturi
Lorraine C. Kevra
Ivor N. Knight
Joseph L. Koepfinger*
D. N. "Jim" Logothetis
L. Bruce McClung
Marco W. Migliaro
Mary Lou Padgett
John W. Pope
Arthur K. Reilly
Gary S. Robinson
Ingo Rüsch
Chee Kiow Tan
Leonard L. Tripp
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
Steve Sharkey
Robert E. Hebner
Chester C. Taylor

Kristin M. Dittmann
IEEE Standards Project Editor

Contents

IEEE Guide for Software Quality Assurance Planning

1. Overview

This guide is divided into six clauses and one annex. Text extracted from IEEE Std 730-1989 is shown in boxes and is used as the introduction to each subclause of the guide as applicable. Clause 1 provides the document overview, scope, and purpose. Clause 2 lists the specific and general references that are useful in applying this guide. Clause 3 describes the contents of each section in a Software Quality Assurance Plan (SQAP) that satisfies IEEE Std 730-1989.[1] Each subclause of clause 3 quotes the applicable wording from the standard using a box to display the quote. Clause 4 provides guidance for implementing an SQAP on a software project, or within a software development organization. Clause 5 provides guidance for evaluating the contents and the implementation of an SQAP. Clause 6 provides guidance for the procedures used to modify an existing SQAP. The annex presents a summary of the SQAP and related standards. This guide is applicable to the development and maintenance of all software, recognizing that the application of these approaches should be tailored to the specific software item. The user of this guide should be aware that efforts are under way to revise and provide additional standards and guides. Prior to implementation, a check should be made with the Secretary, IEEE Standards Board, for further detailed guidance in this area.

1.1 Scope

This guide explains and clarifies the contents of each section of an SQAP that satisfies the requirements of IEEE Std 730-1989. The guide supersedes IEEE Std 983-1986 and does not constitute further requirements than those stated in IEEE Std 730-1989. An organization can claim compliance with IEEE Std 730-1989 without following this guide completely.

1.2 Purpose

This guide presents the current consensus of those in the software development and maintenance community with expertise or experience in generating, implementing, evaluating, and modifying an SQAP. The SQAP should describe the plans and activities for the Software Quality Assurance (SQA) staff. The SQA staff observes the development process and reports deficiencies observed in the procedures and the resulting products.

---

[1] Information on references can be found in clause 2.

2. References, definitions, and acronyms

2.1 Specific references

This guide shall be used in conjunction with the following publications:

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI).[2]

IEEE Std 730-1989, IEEE Standard for Software Quality Assurance Plans (ANSI).

IEEE Std 828-1990, IEEE Standard for Software Configuration Management Plans (ANSI).

IEEE Std 829-1983 (Reaff 1991), IEEE Standard for Software Test Documentation (ANSI).

IEEE Std 830-1993, IEEE Recommended Practice for Software Requirements Specifications (ANSI).

IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI).

IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI).

IEEE Std 990-1987 (Reaff 1992), IEEE Recommended Practice for Ada as a Program Design Language (ANSI).

IEEE Std 1008-1987 (Reaff 1993), IEEE Standard for Software Unit Testing (ANSI).

IEEE Std 1012-1986 (Reaff 1992), IEEE Standard for Software Verification and Validation Plans (ANSI).

IEEE Std 1016-1987 (Reaff 1993), IEEE Recommended Practice for Software Design Descriptions (ANSI).

IEEE Std 1028-1988, IEEE Standard for Software Reviews and Audits (ANSI).

IEEE Std 1033-1985, IEEE Recommended Practice for Application of IEEE Std 828 to Nuclear Power Generating Stations (ANSI).[3]

IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management (ANSI).

IEEE Std 1045-1992, IEEE Standard for Software Productivity Metrics (ANSI).

IEEE Std 1058.1-1987 (Reaff 1993), IEEE Standard for Software Project Management Plans (ANSI).

IEEE Std 1061-1992, IEEE Standard for a Software Quality Metrics Methodology (ANSI).

IEEE Std 1063-1987 (Reaff 1993), IEEE Standard for Software User Documentation (ANSI).

IEEE Std 1074-1995, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1209-1992, IEEE Recommended Practice for the Evaluation and Selection of CASE Tools (ANSI).

---

[2]IEEE publications may be obtained from the IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331 or from the Sales Department, American National Standards Institute, 11 West 42nd Street, New York, NY 10036.

[3]IEEE Std 1033-1985 has been withdrawn; it is, however, available in the *Nuclear Power Standards Collection, 1990 Edition*. Archive copies of withdrawn IEEE standards may also be obtained from Global Engineering, 15 Inverness Way East, Englewood, CO 80112-5704, USA, tel. (303) 792-2181.

IEEE Std 1219-1992, IEEE Standard for Software Maintenance (ANSI).

ISO 9000-1: 1994, Quality management and quality assurance standards—Part 1: Guidelines for selection and use.[4]

2.2 Definitions

The definitions listed below establish meaning in the context of this guide. Other definitions can be found in IEEE Std 610.12-1990 and IEEE Std 730-1989.

2.2.1 conventions: Accepted guidelines employed to prescribe a disciplined, uniform approach to providing consistency in a software item, for example, uniform patterns or forms for arranging data.

2.2.2 guide: Document in which alternative approaches to good practice are suggested but no clear-cut recommendations are made.

2.2.3 practice: Recommended approach, employed to prescribe a disciplined, uniform approach to the software life cycle.

2.2.4 standards: Mandatory requirements employed to prescribe a disciplined, uniform approach to software development, maintenance, and operation.

2.2.5 techniques: Technical and managerial procedures used to achieve a given objective.

2.2.6 software tools: Computer programs used to aid in the development, testing, analysis, or maintenance of a computer program or its documentation.

2.2.7 methodology: A comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed.

2.3 Abbreviations and acronyms

The following alphabetical contractions appear within the text of this guide:

CASE	Computer-Aided Software Engineering
CCB	Configuration Control Board
CDR	Critical Design Review
CI	Configuration Item
CM	Configuration Management
COTS	Commercial-Off-The-Shelf
ICD	Interface Control Document
PDR	Preliminary Design Review
QA	Quality Assurance
RVTM	Requirements Verification Traceability Matrix
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCMPR	Software Configuration Management Plan Review
SDD	Software Design Description
SDP	Software Development Plan

[4]ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

SMM	Software Maintenance Manual
SPMP	Software Project Management Plan
SPM	Standards and Procedures Manual
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SRR	Software Requirements Review
SRS	Software Requirements Specifications
SVVP	Software Verification and Validation Plan
SVVPR	Software Verification and Validation Report
SVVR	Software Verification and Validation Repor
TQM	Total Quality Management
UDR	User Documentation Review
V&V	Verification and Validation

3. Contents of a Software Quality Assurance Plan

NOTE—Original footnotes to the text of IEEE Std 730-1989 appear with updated references beneath the boxes.

The Software Quality Assurance Plan shall include the sections listed below to be in compliance with this standard. The sections should be ordered in the described sequence. If the sections are not ordered in the described sequence, then a table shall be provided at the end of the SQAP that provides a cross-reference from the lowest numbered subsection of this standard to that portion of the SQAP where that material is provided. If there is no information pertinent to a section, the following shall appear below the section heading, "This section is not applicable to this plan," together with the appropriate reasons for the exclusion.

(1) Purpose
(2) Reference documents
(3) Management
(4) Documentation
(5) Standards, practices, conventions, and metrics
(6) Reviews and audits
(7) Test
(8) Problem reporting and corrective action
(9) Tools, techniques, and methodologies
(10) Code Control
(11) Media Control
(12) Supplier Control
(13) Records collection, maintenance, and retention
(14) Training
(15) Risk Management

Additional sections may be added as required.

Some of the material may appear in other documents. If so, then reference to these documents should be made in the body of the SQAP. In any case, the contents of each section of the plan shall be specified either directly or by reference to another document.

Some of the SQAP required information may be contained in other documents, such as a separate Software Configuration Management Plan (SCMP), Software Development Plan (SDP), Software Project Management Plan (SPMP), or Software Verification and Validation Plan (SVVP). The required information also may be documented in approved standards and accepted conventions, practices, or procedures. The sections of the SQAP should reference the documents in which the information is contained. The referenced documents should be reviewed to ensure that they provide all the required information.

3.1 Purpose

3.1 Purpose (Section 1 of the SQAP). This section shall delineate the specific purpose and scope of the particular SQAP. It shall list the name(s) of the software items covered by the SQAP and the intended use of the software. It shall state the portion of the software life cycle covered by the SQAP for each software item specified.

The following questions should be addressed in this section:

a) *What is the intended use of the software covered by this SQAP?* How is the software to be used? How critical is this software? Is it part of a larger system? If so, how is it related to the system?

b) *What is the scope of this SQAP?* Who does it apply to? Who is the intended audience?

c) *Why is this SQAP being written?* Is this plan being written in response to an internal (e.g., in-house goals) or external (e.g., legal or contractual) requirement? How will this plan contribute to the success of the project?

d) *Which software items are covered by this SQAP?* Specific names and abbreviations should be supplied for these items.

e) *Which portions of the software life cycle apply to each of the software items mentioned in 3.1a)?* Name the life cycle model to be used by this project. If the reader of the SQAP is unfamiliar with that name, then refer to the document or documents that define the model. For projects enhancing, modifying, and correcting existing products, identify the life cycle stages or phases they will pass through before being integrated into the existing product. Depending upon complexity of the relationship of the software items to the portions of the software life cycle,[5] a matrix may be provided or referenced.

f) *Why were the documents that form the basis of this SQAP chosen?* Describe the extent to which this SQAP is based on IEEE Std 730-1989 and other documents.

g) *What, if any, are the deviations from the documents mentioned in 3.1e)?* Which software items warrant additional, more rigorous, or more relaxed practices or procedures? Record any deviations that reflect the criticality presented in 3.1a). Justify here, or reference a document with the full justification in terms the customer and developers can understand.

3.2 Reference documents

3.2 Reference Documents (Section 2 of the SQAP). This section shall provide a complete list of documents referenced elsewhere in the text of the SQAP.

Documents used to develop the SQAP should be referenced in the text of the SQAP (e.g., military, industry-specific, or corporate quality assurance standards and guidelines).

By definition, these documents originate outside the project. Some may have confidentiality restrictions so only part of the document is available to the project. Some may have different update and version release procedures. Some may be on paper and some electronic. Some may be located with the project and some may be located elsewhere. Identify any special arrangement to obtain the document and to ensure the project uses the most current official version.

[5] See IEEE Std 1074-1995.

3.3 Management

3.3 Management (Section 3 of the SQAP). This section shall describe organization, tasks, and responsibilities.

Section 3.3.1 shall describe each major element of the organization that influences the quality of the software. Section 3.3.2 shall list the tasks covered by this plan. Section 3.3.3 shall identify specific organizational responsibilities for each task. This description also should identify the management position that retains overall authority and responsibility for software quality.

3.3.1 Organization

3.3.1 Organization. This paragraph shall depict the organizational structure that influences and controls the quality of the software. This shall include a description of each major element of the organization together with the delegated responsibilities. Organizational dependence or independence of the elements responsible for SQA from those responsible for software development and use shall be clearly described or depicted.

The organizational element(s) responsible for the software quality assurance functions covered by the SQAP may be developers knowledgeable in quality assurance tools, techniques, and methodologies; a dedicated quality assurance element serving a number of projects; or a series of separate organizational elements, each of which implements one or more SQA functional activities. The SQAP should state the organizational and functional boundaries of the SQA organizational element. The SQA element can be distributed over different organizational elements. The SQAP should endeavor to identify all. The relationship between the primary SQA organizational element and the software development element should be delineated and explained. This should not be construed to indicate that a specific SQA organization must be established. The most effective organization is a separate Quality Assurance (QA) team that is responsible to an SQA organization rather than to the manager of the software development organization. SQA independence is necessary because the QA manager must not have development responsibilities that tend to override quality concerns.

Customer management, customer and developer contracts departments, the hands-on operational user, and any independent organization contracted for a specific aspect of the project (e.g., training, technical writers, internal verification and validation, or document publishing) contribute their part with the SQA organizational element and developers in the quality of the deliverables. SQA organizational elements should share quality evaluation records and related information with customer organizations to promote resolution of quality issues.

A pictorial organizational structure should be included with an explanation describing the nature and degree of relationships with all organizational elements responsible for software quality and development. The explanation should include the following:

a) A description of each element that interacts with the SQA element.
b) The organizational element delegating authority and delegated responsibilities of interacting elements.
c) Reporting relationships among the interacting elements identifying dependence/independence.
d) Identification of the organizational element with product release authority.
e) Identification of the organizational element or elements that approve the SQAP.
f) The reporting lines for escalating conflicts and the method by which conflicts are to be resolved among the elements.

The explanations also may include

g) The size of the SQA element and the amount of effort dedicated to the project, where the amount is less than 100%.
h) An explanation of any deviations from the organizational structure outlined in existing SQA policies, procedures, or standards.

The description of the organizational structure should be complete so that all the tasks addressed in the SQAP can be directly related to a responsible organization.

3.3.2 Tasks

> **3.3.2 Tasks.** This paragraph shall describe (a) that portion of the software life cycle covered by the SQAP, (b) the tasks to be performed with special emphasis on software quality assurance activities, and (c) the relationships between these tasks and the planned major check-points. The sequence of the tasks shall be indicated.

The SQA tasks are described in 3.4 through 3.15 of the IEEE Std 730-1989 SQAP. Some of these tasks consist of planning activities while others, such as reviews and tests, are directed towards the software product. All the tasks in these sections may not be applicable to a specific project, in which event they may be omitted from the project SQAP. Any omissions or deviations from IEEE Std 730-1989 should be explained in the appropriate section of the project SQAP. Any additional tasks should be included in the appropriate SQAP sections. Any deviations from corporate software quality assurance policies should be explained in the appropriate SQAP sections.

This section of the SQAP also should identify the tasks associated with the publication, distribution, maintenance, and implementation of the SQAP.

All tasks discussed in 3.4–3.15 are SQA tasks and are to be performed by the SQA function. Some of these tasks have a profound influence on the development element, and such tasks should be performed by, or in close cooperation with, the software development element. This is especially true for the tasks defined by 3.4 and 3.5.

Each task identified should be defined together with entrance and exit criteria required to initiate and terminate the task. The output of each task should be defined in such a way that its achievement or termination can be objectively determined in a prescribed manner. Additionally, this section could include a table indicating the staffing levels required for the listed tasks.

While it is strongly recommended that a Software Development Plan (SDP) or a Software Project Management Plan (SPMP) (see 3.4.3.1 or 3.4.3.3 of this guide, respectively) be prepared, if either document is not available, this section should provide schedule information outlining the development cycle to include the software quality assurance activities. This schedule should define the sequence of tasks to be performed, the major milestones occurring during the development life cycle, and the tasks and deliverables that should be completed prior to each major milestone.[6]

3.3.3 Responsibilities

> **3.3.3 Responsibilities.** This paragraph shall identify the specific organizational elements responsible for each task.

If two or more organizational elements share responsibility for a task, their respective responsibilities should be identified. Describe the procedure for resolving issues between organizational elements sharing responsi-

[6]See IEEE Std 1074-1995.

bilities. The management position accountable for overall software quality should be identified. This section of the SQAP should indicate the review and approval cycle, indicating signature authority as required. It should show the number of controlled copies and describe the method of control. It should designate the personnel and organizational element responsible for distributing the SQAP and describe the methods and responsibilities for the approval, distribution, and incorporation of changes (all changes should follow the approved CM procedures). The SQAP should identify the organizational elements responsible for the origination, review, verification, approval, maintenance, and control of the required task documentation as described in 3.4.

A tabular listing or matrix that provides the individual or organizational element responsible for each task described in the SQAP should be provided, including those tasks that are shared. The size and complexity of the project may require the duplication of this information in other plans, especially in the Software Development Plan and the Software Project Management Plan. If duplication exists, reference all documents with duplicate data in order to promote ease of maintenance. In these cases, the data in a responsibility matrix is particularly helpful in clarifying task responsibilities for all aspects of the software project.

3.4 Documentation

3.4 Documentation (Section 4 of the SQAP)

3.4.1 Purpose. This section shall perform the following functions:

(1) Identify the documentation governing the development, verification and validation, use, and maintenance of the software.

(2) State how the documents are to be checked for adequacy. This shall include the criteria and the identification of the review or audit by which the adequacy of each document shall be confirmed, with reference to Section 6 of the SQAP.

3.4.1 Purpose

Section 6 of the SQAP is discussed in 3.6 of this guide.

The SQAP shall identify documentation, whether hardcopy or softcopy, that will be prepared during the development, verification and validation, use, and maintenance of the software. If there is no independent verification and validation, then the quality assurance procedures that are to be used on the project should be identified. Also, all required test documentation should be noted. The SQAP also shall identify the specific reviews, audits, and associated criteria required for each document, including references as appropriate to 3.6, Reviews and Audits.

The following clauses of this guide should describe the format and content of each of the documents used. If this information is provided in another document, only a reference should be given. Organizational policies, procedures, and standards may determine additional information requirements.

3.4.2 Minimum documentation requirements

3.4.2 Minimum Documentation Requirements. To ensure that the implementation of the software satisfies requirements, the following documentation is required as a minimum.

IEEE Std 730-1989 requires the following documents:

a) Software Requirements Specifications (SRS)
b) Software Design Description (SDD)
c) Software Verification and Validation Plan (SVVP)

d) Software Verification and Validation Report (SVVR)
e) User documentation
f) Software Configuration Management Plan (SCMP)

This minimum set of documents has been found to be a solid foundation to assure the quality of a software development in supplier-purchaser relationships. These documents are equally usable in in-house developments with an informal contract or with no contract. They are applicable where the customer is the company marketing division representing the marketplace or the business manager reaching for help to achieve business objectives. Other documentation, such as test plans and database design information, should be provided as applicable. Database design information may be incorporated into the SDD if the project is small in size and the database is not a complex structure.

The QA activities associated with each document review must be scheduled in consonance with the development life cycle phases of the project.

Where the development project changes existing software, the required set of documents may be a subset of the documents used for the original development. A brief description of each document follows.

3.4.2.1 Software Requirements Specifications (SRS)

3.4.2.1 Software Requirements Specification (SRS). The SRS shall clearly and precisely describe each of the essential requirements (functions, performances, design constraints, and attributes) of the software and the external interfaces. Each requirement shall be defined such that its achievement is capable of being objectively verified and validated by a prescribed method; for example, inspection, analysis, demonstration, or test.[a]

[a]See IEEE Std 830-1993.

The SRS is usually developed from one or more documents, such as a user requirements statement, operational requirements, preliminary hazard analysis, software product market definition, parent or previous SRS reverse engineering documentation, system-level requirements and design documentation, statement of work, or contract. It specifies in detail the requirements as agreed upon by the developer and the requester or user. The SQAP should identify the governing documents and state the precedence in the event of two or more documents containing contradictory requirements. Where the SRS is produced by the developer, the SQAP should identify what standards or guides apply to the content and format of the SRS as well as the process used to develop it.[7] The SRS may contain a list of references that identify working models (such as prototypes or products) that are part of the requirements and need to be included in the design (software and system), development, testing and operational phases. The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SRS.

The SRS is subject to the Software Requirements Review (SRR) described in 3.6.2.1, which identifies the SQA organizational element's QA activities.

[7]See IEEE Std 830-1993 for further guidance on the content of an SRS.

3.4.2.2 Software Design Description (SDD)

3.4.2.2 Software Design Description (SDD). The SDD shall depict how the software will be structured to satisfy the requirements in the SRS. The SDD shall describe the components and subcomponents of the software design, including data bases and internal interfaces. The SDD shall be prepared first as the Preliminary SDD (also referred to as the Top-Level SDD) and shall be subsequently expanded to produce the Detailed SDD.[a]

[a]See IEEE Std 1016-1987 (Reaff 1993).

The SDD is a technical description of how the software will meet the requirements set forth in the SRS. Its most important function is to describe a decomposition of the whole system into components (subsystems, segments, etc.) that are complete and well-bounded. In addition, it should document the rationale for the more important design decisions in order to facilitate the understanding of the system structure.

The SDD describes major system features such as data bases, diagnostics, external and internal interfaces, as well as the overall structure of the design. It involves descriptions of the operating environment, timing, system throughput, tables, sizing, centralized or distributed processing, extent of parallelism, client/server, reusable objects library, program design language (PDL), prototypes, modeling, simulation, etc. The SQAP should identify the standards and conventions that apply to the content and format of the SDD, as well as the process and procedures to be used in developing the SDD. If prototyping, modeling, or simulations are used, the SQA organizational element could observe a demonstration, which is a more efficient way to review and assess written design documentation.

Where the SDD is produced by separate teams, the procedures shall relate to the responsibilities in 3.3.3. The SDD may be an evolving document or a document that is baselined after each significant review. A new version containing a more detailed design description is developed for each subsequent review. The SQAP should identify the number and purpose of the SDD documents. Where one document spawns several others at a more detailed level (e.g., the data base design distinct from the user interface design, distinct from the algorithmic processing) this section should explain the scope and precedence of the sibling documents.

The SDD is subject to the Preliminary Design Review (PDR) and the Critical Design Review (CDR), described in 3.6.2.2 and 3.6.2.3, respectively, which identify the SQA organizational element's QA activities, or equivalent design reviews in the chosen life cycle model.

In addition to the design descriptions noted in IEEE Std 1016-1987 (Reaff 1993), which are applicable to 4GL and object-oriented methodologies, the SDD also should consist of items pertaining to earlier languages and methodologies (e.g., 2GL, 3GL, COBOL, and FORTRAN), if applicable, for each component in the system. The SDD should consist of items such as the following:

a) A textual description of the component's
    1) Inputs
    2) Outputs
    3) Calling sequence
    4) Function or task
    5) Algorithms
b) A list of other components called.
c) A list of all calling components.
d) Allowed and tolerable range of values for all inputs.
e) Allowed and expected range of values for all outputs.
f) Assumptions, limitations, and impacts.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SDD.

3.4.2.3 Software verification and validation plan (SVVP)

3.4.2.3 Software Verification and Validation Plan (SVVP). The SVVP shall identify and describe the methods (for example, inspection, analysis, demonstration, or test) to be used:[a]

(1) To verify that (a) the requirements in the SRS have been approved by an appropriate authority, (b) the requirements in the SRS are implemented in the design expressed in the SDD; and (c) the design expressed in the SDD is implemented in the code.

(2) To validate that the code, when executed, complies with the requirements expressed in the SRS.

[a]See IEEE Std 829-1983 (Reaff 1991), IEEE Std 1008-1987 (Reaff 1993), and IEEE Std 1012-1986 (Reaff 1992).

The SVVP describes the overall plan for the verification and validation of the software and could be produced and reviewed incrementally. The tasks, methods, and criteria for verification and validation are described. The SVVP might be used for documentation of the testing standards and practices as they are defined in 3.5.2 and 3.5.2.4. In the same way, it might be used to document the procedures to be followed for some of the reviews defined in 3.6. This section should explain the scope of validation testing to ensure the baselined requirements and explain the stages of development that require customer review and the extent of the verification that will precede such a review. The SVVP specifies minimum test documentation requirements. IEEE Std 829-1983 (Reaff 1991) may be consulted.

The SQAP should identify which standards and conventions apply to the content and format of the SVVP. A stand-alone section of the SVVP should include a verification matrix where requirements are listed with their corresponding SVVP section. This matrix, which is dynamic (and must be maintained), is a tool for the SQA staff to use to verify that all system software requirements have been met. The requirements should be mapped to test cases. The SQA organization should audit the matrix for completion (by all the baselined requirements and tests).

The contents of the SVVP will be evaluated at the Software Verification and Validation Plan Review (SVVPR) described in 3.6.2.4.

3.4.2.4 Software Verification and Validation Report (SVVR)

3.4.2.4 Software Verification and Validation Report (SVVR). The SVVR shall describe the results of the execution of the SVVP.

The SVVR summarizes the observed status of the software as a result of the execution of the SVVP. The SVVR should include the following information:

a) Summary of all life cycle V&V tasks.

b) Summary of task results.

c) Summary of anomalies and resolutions.

d) Assessment of overall software quality.

e) Summary from the verification matrix.

f) Recommendations such as whether the software is, or is not, ready for operational use.

The report may be a full report or a subset, such as a certificate, with the limiting extract being a “tested slip” as seen in some customer products. This section should explain why the style of report was chosen and in what way it satisfies the criticality of the product and gives assurances to the customer.

The SQAP should clearly define the methods to be used by the SQA organizational element to assure the correctness and completeness of the data in the SVVR.

3.4.2.5 User documentation

> **3.4.2.5 User Documentation.** User documentation (e.g., manual, guide, etc.) shall specify and describe the required data and control inputs, input sequences, options, program limitations, and other activities or items necessary for successful execution of the software. All error messages shall be identified and corrective actions described. A method of describing user-identified errors or problems to the developer or owner of the software shall be described. (Embedded software that has no direct user interaction has no need for user documentation and is therefore exempted from this requirement.).[a]

[a]See IEEE Std 1063-1987 (Reaff 1993).

The user documentation section of the SQAP (e.g., documents, videotapes, on-line graphic storyboards, and tutorials) should describe the software’s operational use and be comprised of the following items:

a) User instructions that contain an introduction, a description of the user’s interaction with the system (in the user’s terminology), and a description of any required training for using the system (see also Training manual, 3.4.4.6, in this guide).
b) An overview of the system, its purpose, and description.
c) Input/output specifications.
d) Samples of original source documents and examples of all input formats (forms or displays).
e) Samples of all outputs (forms, reports, or displays).
f) Instructions for data preparation, data keying, data verification, data proofing, and error correction. Wherever software is capable of damage to user assets (e.g., data base contents) the user should be forewarned to avoid accidental damage.
g) References to all documents or manuals intended for use by the users.
h) A description of the system’s limitations.
i) A description of all the error messages that may be encountered, together with recommended steps to recover from each error.
j) Procedures for reporting and handling problems encountered during the use of a software item.
k) Menuing hierarchy and navigation methods.
l) User administration activities (backup, recovery, batch initiation, access control).

There may be several different types of users (e.g., new; those requiring refresher instructions; salespersons; or maintainers). Different user guides or manuals may be required to suit the intended audiences. In each case, the documentation should define the extent of the material delivered to enable the users to display the software.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the user documentation and the process used to develop it.

A User Documentation Review (UDR) is described in 3.6.3.1 of this guide, which identifies the SQA organizational element’s QA activities.

3.4.2.6 Software Configuration Management Plan (SCMP)

3.4.2.6 Software Configuration Management Plan (SCMP). The SCMP shall document methods to be used for identifying software items, controlling and implementing changes, and recording and reporting change implementation status.[a]

[a]See IEEE Std 828-1990 and IEEE Std 1042-1987 (Reaff 1993). See also IEEE Std 1033-1985.

The SCMP of the SQAP should describe the tasks, methodology, and tools required to assure that adequate Software Configuration Management (SCM) procedures and controls are documented and are being implemented correctly. If the SCMP is not a stand-alone document, and is included in the SQAP, it is not necessary that the SQA organizational element prepare the Software Configuration Management Plan (SCMP); however, it is essential that one exist for each project. It is suggested that the CM organizational element prepare the SCMP so that the SQA organizational element can maintain its independence for CM evaluation purposes.

The SQAP should define the extent to which the project requires configuration management. Where the project is part of a larger system that is employing configuration management, this clause should define how the software CM plan fits with the system CM plan.

The SCMP should describe the methods to be used for

a) Identifying the software configuration items.
b) Controlling and implementing changes.
c) Recording and reporting change and problem reports implementation status.
d) Conducting configuration audits.
e) Identifying review and approval cycle as well as signature authority.
f) Identifying the personnel responsible for maintaining the baselines and distributing the SCMP.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SCMP and the process used to develop it.

3.4.3 Other documentation

3.4.3 Other. Other documentation may include the following:
(1) Software Development Plan
(2) Standards and Procedures Manual
(3) Software Project Management Plan
(4) Software Maintenance Manual.

3.4.3.1 Software Development Plan (SDP)

The SDP can be used as the highest-level planning document governing a project, or could be subordinate within a larger set of plans. For example, several SDPs may be written in support of a larger project that is governed by an SPMP. The SDP should identify all technical and managerial activities associated with software development. The SDP should specify the following items, which should be reviewed and assessed by the SQA organizational element:

a) Description of software development.
b) Software development organization responsibilities and interfaces.
c) Process for managing the software development (including allocation of resources, project control mechanisms, software metrics, and risk management).
d) Technical methods, tools, and techniques to be used in support of the software development.

e) Assignment of responsibility for each software development activity.
f) Schedule and interrelationships among activities.
g) Formal qualification testing organization and approach.
h) Software product evaluation during each life cycle phase including subcontractor products.

When multiple SDPs are required, they should focus on items a) and d). The associated SPMP should cover items b), c), e), f), g), and h) in order to provide a unified approach to design, coding, and testing.

The SQAP should define the procedures for creating the data in the SDP and criteria for updating and assuring its quality. Any deviations from the plan should be reconciled between the SQA staff and the software development manager. The plan should be updated and the latest version clearly identified. Obsolete versions should be maintained according to the organization's policy. Procedures should be defined for the approval and distribution of the SDP.

3.4.3.2 Standards and Procedures Manual (SPM)

The SPM should provide details on all project standards and procedures to be followed. At a minimum, the information described in 3.5 should be included.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SPM.

3.4.3.3 Software Project Management Plan (SPMP)

The SPMP can be used in place of an SDP, or as a plan that governs a larger project that has subordinate projects each covered by SDPs. The SPMP should identify all technical and managerial activities associated with an instance of software development.[8] The SPMP should specify the following items, which should be reviewed and assessed by the SQA organizational element:

a) Description of software development (high-level description if details are contained in an SDP. If there is no SDP, full details are required in the SPMP).
b) Software development and management organizations responsibilities and interfaces.
c) Process for managing the software development (including allocation of resources, project control mechanisms, software metrics, and risk management).
d) Technical methods, tools, and techniques to be used in support of the software development (high-level description if details are provided in SDP).
e) Assignment of responsibility for each activity (e.g., specific skill assignments, key personnel).
f) Schedule and interrelationships among activities.
g) Process improvement activities.
h) Goals deployment activities.
i) Strategic quality planning efforts triggered by reviews (e.g., post mortem), and benchmarking.
j) A list of deliverables.
k) Subcontractor(s) project management plan(s).

The SQAP should define the procedures for creating the data in the SPMP and criteria for updating and assuring its quality. Any deviations from the plan should be reconciled between the SQA staff and the software project manager. The plan should be updated and the latest version clearly identified. Obsolete versions should be maintained according to the organization's policy. Procedures should be defined for the approval and distribution of the SPMP.

[8]See IEEE Std 1058.1-1987 (Reaff 1993).

3.4.3.4 Software Maintenance Manual (SMM)

A maintenance manual should contain instructions for software product support and maintenance, such as procedures for correcting defects, installation of enhancements, and testing of all changes.[9] All hardware and software configuration specifications required to maintain the software should be described in detail. Any unusual settings or known anomalies should be identified in order to aid in efficient maintenance. New versions of software should be thoroughly tested prior to incorporation into operational systems. Version control procedures should be reviewed and approved by SQA and SCM organizational elements. The SQA organizational element should periodically audit and validate the use of the version control procedures as well as the software maintenance process and procedures. The SMM should refer to the Problem Reporting System (3.8) and the SCMP (3.4.2.6) in this guide.

3.4.4 Additional suggested documentation

The attributes, context, and environment of the product could dictate inclusion of additional documents, such as, but not limited to, the following:

a) User requirements statement
b) External interface specifications
c) Internal interface specifications
d) Operations manual
e) Installation manual
f) Training manual
g) Training plan
h) Software metrics plan
i) Software security plan

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the documents described in the following subclauses (3.4.4.1–3.4.4.9).

3.4.4.1 User requirements statement

A user requirements statement can be used as a high-level document preceding the approved SRS for a large development, in place of an SRS in cases where minor changes are made to an operational system that has no SRS, or as a means of passing requirements on to a supplier. The user requirements statement should include, but is not limited to

a) A natural language service request that contains the identity of the requester, the software item name and title, the date the software item was requested and is required, a description of what the software item should do, an abstract of the need for the software item, privacy or security considerations, and a list of potential users of the software item.
b) A list of the objectives that are to be satisfied by the software item, as well as any other needs (e.g., administrative, timing, quality) and restraints the user perceives as necessary.
c) References to and summarized conclusions from any studies done to define resource requirements (e.g., hardware, software, personnel, plant and facilities, or environmental), feasibility, or cost-benefit analyses.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the user requirements statement.

[9]See IEEE Std 1219-1992.

3.4.4.2 External interface specifications

External interface specifications should be contained within the software requirements specifications, the system design document, or an Interface Control Document (ICD). In situations where the detailed external interface specifications are not available in the design documentation or ICD, a separate external interface specifications document may be required that would provide lower-level detail.

The external interface specifications should contain information about files and other connections, such as messages passed, to software items outside the system under development. Consideration should be given to human interfaces, hardware interfaces, environmental constraints (such as weather conditions) and data structures and files or transactions coming from or going to other systems, standards, protocols, timing issues, and throughput or capacity of the interfaces.

3.4.4.3 Internal interface specifications

Internal interface specifications are a subset of the design documentation and may be traced to a software requirement identified in the SRS. In situations where the internal interface specifications are not available in the design documentation, and/or an ICD, a separate internal interface specifications document may be required.

The internal interface specifications should contain information about files and other connections among all the components within the system. Consideration should be given to such subjects as transfer of control between modules, passing of data between modules, physical and logical interfaces, common data, timing and concurrence management techniques.

3.4.4.4 Operations manual

The operations manual should contain at least the following items:

a) Operating instructions that include
   1) An introduction
   2) Run schedules
   3) Setup requirements/procedures
   4) Run control procedures
   5) Error procedures
   6) Security procedures
   7) Distribution procedures
   8) Backup and recovery procedures
   9) Restart procedures
   10) Termination procedures
   11) Tutorial and practice procedures
b) Specifications for the system, including environmental requirements
c) Input/output specifications
d) Auditing controls

3.4.4.5 Installation manual

An installation manual should contain instructions for the installation of the software, for example, file conversion instructions, use of user-controlled installation options, and instructions for performing an installation test. Installation procedures may be performed through an interactive interface (i.e., menu driven).

3.4.4.6 Training manual

The training manual should contain information necessary for training users and operators of the system. It should contain, but is not limited to, the following:

a) An introduction
b) How to use the system
c) How to prepare input
d) Data input descriptions
e) Data control descriptions
f) How to run the system
g) Output distributions
h) Description of output data and interpretations (e.g., error messages)
i) Tutorial and practice exercises
j) How to get help

3.4.4.7 Training plan

The development of software products that require complex or unfamiliar interactions with users and operators should include a comprehensive plan for training. The training plan should include the following:

a) A description of the populations to be trained, the training objectives for each population, and the content to be covered in the training.
b) An estimate of the amount of resources necessary for training development, delivery, and time expenditures.
c) Procedures for evaluating the effectiveness of the training and for making modifications to the training.

3.4.4.8 Software metrics plan

The software metrics plan should address the way product and process metrics will be used to manage the development, delivery, and maintenance processes. (See 3.5.2.7.) The software metrics plan should contain information on the following:

a) The mission and objectives of the software metrics program.
b) The quantitative measures (target levels and acceptable ranges if applicable) of the quality of the software products.[10]
c) How the product or process metrics will be used to identify and measure performance.
d) How remedial action will be taken if product or process metric levels grow worse or exceed established target levels.
e) Improvement goals in terms of the product and process metrics.
f) The quantitative measures (target levels and acceptable ranges, if applicable) of the quality of the software processes.[11]
g) How the metrics will be used to determine how well the development process is being carried out in terms of milestones and in-process quality objectives being met on schedule.
h) How the metrics will be used to determine how effective the development process improvement is at reducing the probability that faults are introduced or that any faults introduced go undetected.
i) Data collection methodology including roles and responsibilities, retention, and validation.
j) A definition of metrics reports that are generated, including their frequency, reporting periods, as well as which element in the software organization uses these reports.
k) How to validate the software quality metric.

[10] See IEEE Std 981.1-1988 and 981.2-1988.
[11] See IEEE Std 981.1-1988 and 981.2-1988.

3.4.4.9 Software security plan

The software security plan should address the way in which the software and the data will be protected from unauthorized access or damage. The software security plan should contain information on the following:

a) How the data should be classified and how this will be communicated (e.g., “no trespassing” messages).
b) How the users of the software access the application and how that access is to be controlled.
c) Network design.
d) User identifications, passwords, security logging, and auditing.
e) Super-user password control and protection of path through the system administrator.
f) Physical security.
g) Virus protection.
h) How employees will be trained on security procedures and practices.
i) The method by which this security plan will be audited for compliance.
j) Disaster plan.
k) Whether the system provides file encryption and decryption capability.

3.5 Standards, practices, conventions and metrics

3.5 Standards, Practices, Conventions and Metrics (Section 5 of the SQAP).

3.5.1 Purpose

3.5.1 Purpose. This section shall:
(1) Identify the standards, practices, conventions and metrics to be applied.
(2) State how compliance with these items is to be monitored and assured.

This section of the SQAP shall identify the standards (mandatory requirements), practices (recommended approach), conventions (accepted guidelines), and metrics (system of measurement) to be employed by all associated with the project, including management and vendors. It should specify the phases of the life cycle to which they apply. Also, it shall specify how compliance will be monitored and assured (see 3.6).

3.5.2 Content

3.5.2 Content. The subjects covered shall include the basic technical, design, and programming activities involved, such as documentation, variable and module naming, programming, inspection, and testing. As a minimum, the following information shall be provided:[a]
(1) Documentation standards
(2) Logic structure standards
(3) Coding standards
(4) Commentary standards
(5) Testing standards and practices
(6) Selected software quality assurance product and process metrics such as:
    (a) Branch metric
    (b) Decision point metric
    (c) Domain metric
    (d) Error message metric
    (e) Requirements demonstration metric.

[a]See IEEE Std 990-1987 (Reaff 1992), IEEE Std 982.1-1988, and IEEE Std 982.2-1988.

The SQAP should identify or reference the standards, practices, conventions, and metrics to be used on the project. As a minimum, the information required by IEEE Std 730-1989 should be addressed in the following life-cycle phases: software requirements, design, implementation, testing, and maintenance. In addition, the standards, practices, and conventions pertaining to software documentation and the use of metrics should be addressed. It is strongly recommended that these issues be resolved in close cooperation with the software development element. The descriptions of the standards, practices, conventions, and metrics are often given in a standards and procedures manual (see 3.4.3.2) or in an SDP. Use of all of the above should state how compliance is assured.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data discussed in the following subclauses (3.5.2.1–3.5.2.7).

3.5.2.1 Requirements phase

Identify or reference the standards, practices, conventions, and metrics to be used during the requirements phase. Cite any internal (e.g., project or corporate) or external (e.g., military or contractual) standards with which requirements baselining and traceability must comply. Use formal requirements statement languages, either textual or graphic, whenever possible.[12] Provision should be made for a scheme that uniquely identifies each requirement. This facilitates traceability and compliance during the subsequent phases.

3.5.2.2 Design phase

Identify or reference the standards, practices, conventions, and metrics to be used during the preliminary design phase where the overall structure of the software system is defined. Cite any internal (e.g., project or corporate) or external (e.g., military or contractual) standards with which the design baselining must comply. Top-down design, which is the most acceptable methodology, should be used whenever feasible. Use of graphic techniques and computer-aided software engineering (CASE) tools that aid in compliance verification should be considered. Object-oriented design methodology should be used with appropriately sized hardware and software in those situations where a simpler design is used.

For the detailed design phase, state what standards, practices, conventions, and metrics will be used for specifying the internal structure of each program module and the interfaces among them. Address such matters as naming conventions and argument list standards. Give serious consideration to requiring the use of a program design language.

3.5.2.3 Implementation phase

Identify or reference the standards, practice, conventions, and metrics to be used during the implementation phase. Cite any internal (e.g., project or corporate) or external (e.g., military or contractual) standards with which implementation procedures must comply. Address such topics as the end-use computer, programming language(s), module size, declaration statement conventions, naming and labeling conventions, component layout standards, the use of structured coding techniques (or structuring precompilers) and CASE tools. Consider data conversion techniques for new systems that are replacing old ones. Standards for the inclusion of comment statement should be covered here. Use standard support software and software tools whenever possible or state reasons for the use of nonstandard support software and tools.

3.5.2.4 Test phase

Identify or reference the standards, practices, conventions, and metrics to be used during the testing phase. Cite any internal (e.g., project or corporate) or external (e.g., military or contractual) standards with which all levels of testing must comply. This includes unit, integration, system and acceptance testing, as well as regression testing. Identify the test environment, which should be the same as the targeted environment.

[12]See IEEE Std 830-1993.

IEEE Std 829-1983 (Reaff 1991) describes an integrated set of test documents and IEEE Std 1008-1987 (Reaff 1993) defines a systematic approach to unit testing.

Address criteria for test repeatability and test coverage such as testing every requirement, user procedure, and program statement. Specify techniques for tracing the test coverage to the test set. Indicate whether any support software will be required, and state how and from where this software will be obtained. State the method and process for certification of test tools or other support software used for demonstration.

3.5.2.5 Maintenance phase

Identify or reference the standards, practices, conventions, and metrics to be used to maintain the software. Cite any external (e.g., user, customer) requirements or standards with which maintenance practices must comply. Cite any internal standards or conventions, such as those describing design, implementation, test, and documentation requirements, that affect the maintenance process. Specify the methods and techniques (both manual and automated) for controlling and managing the software maintenance process such as requiring formal written change requests, review and evaluation of change requests, prioritizing and scheduling approved change requests, monitoring of the maintenance process through reviews and audits, and the collection and analysis of data affecting and resulting from the maintenance process. IEEE Std 1219-1992 defines a systematic approach to maintenance.

3.5.2.6 Documentation

Identify or reference the standards, practices, conventions, and metrics to be used in preparing and submitting software documentation. Cite any internal (e.g., project, corporate) or external (e.g., military, contractual) standards with which documentation must comply. Cite any document interchange (softcopy) standards that are applicable. The types of documentation that should be addressed may include software plans (e.g., SQAP, SCMP), software development documents (e.g., SRS, SDD), and any software deliverables (e.g., source code, users' manuals, tools).

3.5.2.7 Metrics

Identify or reference the standards, practices, and conventions to be used in the definition, collection and utilizations of software metrics data. Cite any internal (e.g., project, corporate) and external (e.g., user, customer) requirements or standards with which metrics practices must comply. IEEE Std 1045-1992 describes conventions for counting the results of the development processes. IEEE Std 1061-1992 provides a methodology for selecting and implementing process and product metrics. IEEE Std 982.1-1988 and Std 982.2-1988 provide various measures for use in different life cycle phases to gain confidence in the building of reliable software.

3.6 Reviews and audits

3.6 Reviews and Audits (Section 6 of the SQAP)

3.6.1 Purpose. This section shall:[a]

(1) Define the technical and managerial reviews and audits to be conducted.
(2) State how the reviews and audits are to be accomplished.
(3) State what further actions are required and how they are to be implemented and verified.

[a]See IEEE Std 1028-1988.

3.6.1 Purpose

The software items produced during the software life cycle process[13] should be reviewed and audited on a planned basis to determine the extent of progress and to evaluate the technical adequacy of the work and its

conformance to software requirements and standards. Technical reviews and audits should be conducted to evaluate the status and quality of the software development effort. “The examination of project issues (both technical and managerial) occurs at various phases during the project life cycle. The results of such examinations are meant to permit improvement of the methods of ensuring software quality and the ability to meet time and cost constraints. The evaluation of software elements occurs during the construction of the element(s) and on its completion. This ensures that the element(s) completely and correctly embodies its baseline specifications” (IEEE Std 1028-1988, Section 3). Completion of reviews provides assurance that design integrity is maintained, technical deficiencies are identified, and necessary changes have been identified and implemented.

This section of the SQAP should identify the specific technical and managerial reviews and audits to be conducted with respect to the software development plans, schedules, and environment. It should describe the procedures to be used in the conduct of reviews and audits, and it should identify the participants (including vendors) and their specific responsibilities. These review and audit procedures should identify specific responsibility for the preparation of a report upon the completion of each review. This section should identify by position or job title who is to prepare these reports, the report format, who is to receive the reports, and associated management responsibilities. The review and audit procedures should describe the follow-up actions to assure that the recommendations made during the reviews and audits are properly implemented. This section should indicate the interval of time between performance of the review or audit and performance of the follow-up. Also, it should identify those responsible for performing follow-up actions.

3.6.2 Minimum requirements

3.6.2 Minimum Requirements. As a minimum, the following reviews shall be conducted:

a) Software Requirements Review (SRR)
b) Preliminary Design Review (PDR)
c) Critical Design Review (CDR)
d) Software Verification and Validation Plan Review (SVVPR)
e) Functional audit
f) Physical audit
g) In-Process audits
h) Managerial reviews
i) Software Configuration Management Plan Review (SCMPR)
j) Post mortem review

Tailoring or inclusion of additional reviews and audits should be made as local, contractual, or project-specific conditions dictate.

[13]See IEEE Std 1074-1995.

Table 1 presents an example of the relationships and timing of these reviews and audits to the software development phases.

Table 1—Example of relationships and timing of required reviews and audits

Typical software development phases (per IEEE Std 1074-1995)	Required software development products (documentation per 3.4.2 and 3.7)	Required SQA audits and reviews[a] per 3.6.2
Requirements	SQAP and SCMP[b] SRS SVVP[c] SDP	SRR In-Process audit[d] SVVPR[c] Managerial review[d]
Design	Preliminary SDD SDD User documentation	PDR[e] In-Process audit CDR[e] UDR[e]
Implementation	Software items with documentation	In-Process audit
Test	Test documentation[f]	Functional audit
Installation and checkout[g]	Deliverable items and SVVR	Physical audit
Operation and maintenance[h]	Products depend on scope of maintenance. Major modifications will have some or all of the above products.	Review depends on scope of required products.

[a]Results of these activities are reports that identify what was reviewed, the deficiencies found, and conclusions. A report generated by a review meeting also includes recommendations as to what needs to be done to resolve the deficiencies. (The items subject to review are the software development products as well as the SQAP and process.)
[b]This includes any referenced documents.
[c]The SVVP completion and SVVPR should be accomplished prior to the PDR.
[d]In-process audits and managerial reviews are scheduled as required throughout the software life cycle. For additional assistance, see clause 5.
[e]A UDR may be held independently of other reviews or in conjunction with the PDR and the CDR (a UDR is not an IEEE Std 730-1989 requirement).
[f]Refer to IEEE Std 829-1983 (Reaff 1991).
[g]In the event this phase is not used in the SQAP, move the required products and audit to the test phase.
[h]This phase is in addition to typical software development phases to show that the SQA effort can be an iterative process.

3.6.2.1 Software Requirements Review (SRR)

3.6.2.1 Software Requirements Review (SRR). The SRR is held to ensure the adequacy of the requirements stated in the SRS.

The SRR is an evaluation of the Software Requirements Specifications (SRS). The SRR is conducted to ensure the adequacy, technical feasibility, and completeness of the requirements stated in the SRS. The SRR should evaluate the SRS for the attributes required by IEEE Std 830-1993 (unambiguous, complete, verifiable, consistent, modifiable, traceable, and usable during the operation and maintenance phase). The review ensures that sufficient detail is available to complete the software design.

The SQAP should indicate the organizational element responsible for conducting the SRR. All organizational elements that contribute or are impacted by the requirements should participate. These may include software design, software test, software quality assurance, system engineering, customers, users marketing, manufacturing, security, etc.

The SQAP should specify how, during the SRR, the quality of the following items will be assessed:

a) Traceability and completeness of the requirement from the next higher level specification (such as a system specification or user requirements specification).
b) Adequacy of rationale for any derived requirements.
c) Adequacy and completeness of algorithms and equations.
d) Correctness and suitability of logic descriptions that may be warranted.
e) Compatibility of external (hardware and software) interfaces.
f) Adequacy of the description of and approach to the human-machine interface.
g) Consistency in the use of symbols and in the specification of all interfaces.
h) Availability of constants and tables for calculations.
i) Testability of the software requirements.
j) Adequacy and completeness of the verification and acceptance requirements.
k) Completeness and compatibility of interface specification and control documentation.
l) Freedom from unwarranted design detail.

Additional items to be considered for assessment of the SRR include

m) Trade-off and design studies that have applicability for decisions on
    1) Data base design and/or real-time design issues.
    2) Programming language characteristics and usage.
    3) Resource allocation (e.g., storage, machine cycles, I/O channel personnel and hardware).
    4) Operating system or executive design, or both.
    5) Development versus COTS solution.
n) The general description of the size and operating characteristics of all support software (e.g., operational program, maintenance and diagnostic programs, compilers, etc.).
o) A description of requirements for the operation of the software and identification of functional requirements such as functional simulation, performance, environmental recording and analysis, exercise configuration, etc.

The results of the review should be documented in an SRR report that identifies all deficiencies described in the review and provides a plan and schedule for corrective action. Also, a decision about whether or not to proceed should be made based on cost estimates, feasibility studies, and project risk assessments. After the decision to proceed is made and the SRS is updated to correct any deficiencies, it should be placed under configuration control to establish the baseline to be used for the software design effort. The SQA organizational element should use the baselined requirements to develop a Requirements Verification Traceability Matrix (RVTM). This RVTM should be used as a tool by the SQA organizational element to validate satisfaction of requirements in the design, code, test, and functional capabilities of the system being developed.

3.6.2.2 Preliminary Design Review (PDR)

3.6.2.2 Preliminary Design Review (PDR). The PDR (also known as top-level design review) is held to evaluate the technical adequacy of the preliminary design (also known as top-level design) of the software as depicted in the preliminary software design description.

The PDR, the pivotal step in the design phase, is held to evaluate the technical adequacy of the preliminary design, to include architectural design, before the beginning of detailed design. The review assesses the progress, consistency, and technical adequacy of the selected design approach; checks the compatibility of the design with the functional and performance requirements of the SRS; and verifies the existence compatibility of the interfaces between the software, hardware, and end users. The PDR also is conducted to determine that the preliminary SDD defines a suitable software design that fulfills the requirements contained in

the SRS. The RVTM should be updated to include the preliminary design mapped to the requirements and verified by the SQA organizational element.

The SQAP should indicate the organizational element responsible for conducting the PDR. All organizational elements that impose requirements or are impacted by the design should participate in the review. These groups could include system engineering, software development, software test, software quality assurance, the customers, users, etc.

The SQAP should specify how, during the PDR, the quality of the following items will be assessed:

a) All detailed functional interfaces with other software, system equipment, communication systems, etc., for adequate identification of interface design and design solution adequacy.
b) The software design as a whole, emphasizing allocation of software components to functions, functional flows, storage requirements and allocations, software operating sequences, and the design of the data base.
c) The human factor requirements and the human-machine interfaces for adequacy and consistency of design.
d) Testability of the design, such as the existence of data stores and process that support behavior and state determination.
e) Test concepts, requirements, documentation, and tools, for adequacy.

Additional items to be considered for assessment of the PDR include

f) An analysis of the design for compatibility with critical system timing requirements, estimated running times, absence of race conditions and deadlock states, and other performance requirements.
g) Technical accuracy and currency of all available test documentation and its compatibility with the test requirements of the SRS.

The results should be documented in a PDR report that identifies all deficiencies discovered during the review and a plan and schedule for corrective action. The updated preliminary SDD document should be placed under configuration control to establish the baseline for the detailed software design effort.

3.6.2.3 Critical Design Review (CDR)

3.6.2.3 Critical Design Review (CDR). The CDR (also known as detailed design review) is held to determine the acceptability of the detailed software designs as depicted in the detailed software design description in satisfying the requirements of the SRS.

The CDR is an evaluation of the completed Software Design Description (SDD), which includes all PDR updates and should provide the low-level details of the design. The CDR is conducted to evaluate the technical adequacy, completeness, and correctness of the detailed design of the software before the start of formal coding (not to include code associated with proof of concept or rapid prototyping activities) or as referenced to the life cycle described in an SDP. The purpose of the CDR is to evaluate the acceptability of the detailed design, to establish that the detailed design satisfies the requirements of the preliminary SDD and the SRS, to review compatibility with the other software and hardware with which the product is required to interact, and to assess the technical, cost, and schedule risks of the product design. The RVTM should be updated to include the detailed design mapped to preliminary design and requirements and verified by the SQA organizational element.

The SQAP should indicate the organizational element responsible for conducting the CDR. All other organizational elements that impose requirements or are impacted by the design should participate. These groups should include system engineering, software development, software test, software quality assurance, customers, users, etc.

The SQAP should specify how, during the CDR, the quality of the following items will be assessed:

a) The compatibility of the detailed design with the SRS.
b) Available data in the forms of logic diagrams, algorithms, storage allocation charts, and detailed design representation (e.g., flow charts, program design language) to establish design integrity.
c) Compatibility and completeness of interface requirements.
d) All external and internal interfaces, including interactions with the data base.
e) Technical accuracy and currency of all available test documentation and its compatibility with the test requirements of the SRS.
f) The requirements for the support and test software and hardware to be used in the development of the product.
g) The final design, including function flow, timing, sizing, storage requirements, memory maps, data base, and other performance factors.

Additional items to be considered for assessment of the CDR include

h) Final report of any trade-off analysis and design studies.
i) Chosen or recommended design tools and design.

The results of the review should be documented in a CDR report that identifies all deficiencies discovered during the review and a plan and schedule for corrective actions. The updated SDD document, when placed under configuration control, establishes the baseline for coding.

3.6.2.4 Software Verification and Validation Plan Review (SVVPR)

3.6.2.4 Software Verification and Validation Plan Review (SVVPR). The SVVPR is held to evaluate the adequacy and completeness of the verification and validation methods defined in the SVVP.

The SVVPR is an evaluation of the completed Software Verification and Validation Plan (SVVP). Since the SVVP may be developed incrementally, multiple reviews may be required. These reviews are held to assure that the verification and validation methods described in the SVVP are adequate and will provide complete evaluation data.

The SQAP should indicate the organizational element responsible for conducting the Software Verification and Validation Plan Review. All organizational elements that impose requirements or are impacted by the SVVP should participate. These groups could include system engineering, software development, software design, software test, software quality assurance, customers, users, etc.

The SQAP should specify how, during the SVVPR, the quality of the following items will be assessed:

a) All verification and validation methods, along with completion criteria to ensure traceability to, and compatibility with, the functional and performance requirements expressed in the SRS.
b) Reports to adequately documents results of all reviews, audits, and tests based on the requirements listed in the SVVP.
c) Adequate descriptions of the software configuration to be tested, including test support software and hardware.
d) Test plans and test designs to assure that all requirements are tested.
e) Test procedures and test cases to assure that test inputs and success criteria are adequately defined and that test instructions are clear and concise.
f) A test schedule identifying which tests are to be done, when, and by whom.
g) Validation of the requirements listed in the SVVP with the baseline and all updates to the SRR list, which should be documented in the RVTM, should be performed by the SQA organizational element.

An additional item to be considered for assessment of the SVVPR could include

h) If the SVVP was incrementally developed, a review of previous reviews to assess correction of previously noted deficiencies.

The RVTM should be updated to include the tests mapped to the code, design, and requirements and verified by the SQA organizational element.

The results of the review should be documented in an SVVPR report that identifies all deficiencies discovered during the review and provides a plan and schedule for corrective action. The updated SVVP, when placed under configuration control, establishes the baseline for software verification and validation activities.

3.6.2.5 Functional audit

3.6.2.5 Functional Audit. This audit is held prior to the software delivery to verify that all requirements specified in the SRS have been met.

The functional audit compares the software as built (including its executable forms and available documentation) with the software requirements as stated in the baselined SRS. Its purpose is to assure that the code addressed all, and only, the documented requirements and functional capabilities stated in the SRS. The functional capabilities should be added to the RVTM and mapped to the tests, code, design and requirements. The SQA organizational element should validate the updated RVTM by observing a system demonstration.

The SQAP should indicate the organizational element responsible for the functional audit. The results are to be documented in the functional audit report, which identifies all discrepancies found and a plan and schedule for their resolution. Audits are not considered to be complete until all discrepancies have been resolved.

Input to the functional audit should consist of the following:

a) Software Requirements Specifications (SRS).
b) Software Verification and Validation Report (SVVR).
c) Software Verification and Validation Plan Review (SVVPR) report.

3.6.2.6 Physical audit

3.6.2.6 Physical Audit. This audit is held to verify that the software and its documentation are internally consistent and are ready for delivery.

The physical audit compares the code with its supporting documentation. Its purpose is to assure that the documentation to be delivered is consistent and correctly describes the code.

The SQAP should indicate the organizational element responsible for conducting the physical audit. The results of the physical audit are to be documented in the physical audit report, which identifies all discrepancies and the plans for their disposition. Once the plans have been approved and implemented (all dispositions have been completed), the software can be delivered.

Input to the physical audit should consist of the following:

a) Software Design Description (SDD).
b) Software products (e.g., code, algorithms, graphical user interfaces).
c) Associated documentation (e.g., engineering notes, software development files, user documentation).

3.6.2.7 In-Process audits

3.6.2.7 In-Process Audits. In-process audits of a sample of the design are held to verify consistency of the design, including:
(1) Code versus design documentation
(2) Interface specifications (hardware and software)
(3) Design implementation versus functional requirements
(4) Functional requirements versus test descriptions.

In-Process audits of samples of the product development items are held as required by the SQAP. The SQAP should indicate the organizational element responsible for conducting the in-process audits. Software inspections and design and test documentation reviews should be included as part of the in-process audit activity. The objective is to verify the consistency of the product as it evolves through the development process by determining that

a) Hardware and software interfaces are consistent with design requirements in the SRS.

b) The functional requirements of the SRS are fully validated by the SVVP.

c) The design of the product, as the SDD is evolving, satisfies the functional requirements of the SRS.

d) The code is consistent with the SDD.

The RVTM should be updated to include the software units (code modules) mapped to the design and requirements and verified by the SQA organizational element.

The result of all in-process audits are measures of how well the development process is working. They should be documented in in-process audit reports that identify all discrepancies found and the plans and schedules for their resolution.

3.6.2.8 Managerial reviews

3.6.2.8 Managerial Reviews. Managerial reviews are held periodically to assess the execution of all of the actions and the items identified in the SQAP. These reviews shall be held by an organizational element independent of the unit being reviewed, or by a qualified third party. This review may require additional changes in the SQAP itself.

Managerial reviews are assessments of the execution of the SQAP. All SQA activities are evaluated. The planned frequency and structure of the managerial reviews should be stated in the SQAP. They should be conducted at the direction of an appropriate level of management independent of the organization performing the software activity. The independent organization could be appointed by management as a process review team of peers.

A managerial review results in a statement as to the adequacy of the SQAP and its execution. This should be conducted by an independent element. Each review should be documented by a report summarizing the review findings, including any exceptions to the process stated in the SQAP, and any recommended changes or improvements.

Clause 5 provides guidance for evaluating the contents and the implementation of an SQAP.

3.6.2.9 Software Configuration Management Plan Review (SCMPR)

3.6.2.9 Software Configuration Management Plan Review (SCMPR). The SCMPR is held to evaluate the adequacy and completeness of the configuration management methods defined in the SCMP.

This SCMPR is an evaluation of the completed SCMP. The review is held to assure the software configuration management procedures described in the SCMP are adequate and will provide the necessary control over documentation and code. The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data presented at the SCMPR.

The SQAP should indicate the organizational element responsible for conducting the SCMPR. All organizational elements that impose requirements on or are impacted by the SCMP should participate. These groups could include systems engineering, software development, software test, SQA, etc.

The following items could be specified in the SQAP as a checklist for the SCMPR:

a) A description of the SCM's purpose, scope of application, key terms, and references for the project.
b) SCM management information that identifies the responsibilities and authorities for accomplishing the planned activities (*WHO?*).
c) Information on all SCM activities to be performed during the project (*WHAT?*).
d) SCM schedule that identifies the required coordination of SCM activities.
e) Identification of tools, physical, and human resources required for execution of SCM (*HOW?*).
f) Overview description of the specified software life cycle.
g) Identification of the software CIs.
h) Identification of other software items to be included, such as support or test software.
i) Relationship of SCM to the hardware or system CM activities during the software life cycle.
j) Reviews and audits that are applied to life cycle milestones.
k) Assumptions that might have an impact on the cost, schedules, or ability to perform or define SCM activities such as assumption of the degree of customer participation in the SCM activities or the availability of automated aides.
l) Status reports that identify the configuration at any point in time during the software life cycle.

Once the SCMP methods and procedures are validated as complete and adequate, routine and spot checking reviews should be conducted by an independent SQA reviewer (i.e., independent from the development activity) in order to verify the use of the approved methods and procedures. Each review should be documented by a report summarizing the review findings, including any exceptions to the process, and any recommended changes or improvements. The implementation of all recommended changes should be monitored and the SCMP methods and procedures should be reevaluated once implementation is complete. This would promote the baselining of standard CM procedures, which is an efficient way to conduct business. New program specific requirements may create the need for the tailoring of the SCMP procedures and methods. The size, scope, and complexity of each program is different; therefore, careful in-depth review should be attempted and the SCMP methods and procedures should be updated and rebaselined as appropriate.

3.6.2.10 Post mortem review

3.6.2.10 Post Mortem Review. This review is held at the conclusion of the project to assess the development activities implemented on that project and to provide recommendations for appropriate action.

Upon conclusion of the development project, a non-attribution, lessons-learned (post mortem) review should be conducted. The objective of the post mortem should be to improve the development process and proce-

dures, not to blame or reward the development team. The SQAP should clearly define the methods used by the SQA organizational element to verify and validate the data presented at the post mortem review.

Independent objective coordinators should gather all of the inputs. Inputs could be obtained anonymously on written forms (either blank or questionnaires) and/or during an evaluation meeting. All who had a part in the development (full life cycle—concept through operation) should be asked their opinion. The scope and size of the project as well as resources available determine how many personnel could participate; however, a representative from each phase or group should give input. Technical personnel, staff [Quality Assurance/ Configuration Management (QA/CM)] administration support managers and, if possible, the users of the system should provide comments on problem areas as well as good practices. Whenever possible, all should discuss their observations and make recommendations for correction to the process. A written summary of the observations as well as the recommendations should be documented by the independent evaluators. A system for reporting problems could be based on that described in 3.8; however, the responsibilities may differ between process and software problems. Also refer to 3.13.1b) of this guide on records collection. Corrections to the standards and operating procedures should be implemented during the next development cycle and an appropriate metric (means to evaluate improvement) should be chosen.

3.6.3 Other

3.6.3 Other. Other reviews and audits may include the user documentation review (UDR). This review is held to evaluate the adequacy (e.g., completeness, clarity, correctness, and usability) of user documentation.

Other reviews may be conducted as required by the organizations responsible for the development. The scope and size of the projects may warrant the additional reviews [e.g., a large user community may require a UDR, the Total Quality Management (TQM) organization may require a Quality Assurance Audit].

3.6.3.1 User Documentation Review (UDR)

The UDR is held to determine the technical adequacy of the documentation approach and design as described in draft versions of the user documentation (softcopy or hardcopy).

The SQAP should indicate the organizational element responsible for conducting the UDR. All organizational elements that are affected or impacted by the user documentation should participate in the review. These groups may include system engineering, software development, software test, software quality assurance, customers, users, etc.

The following items should be specified in the SQAP as the UDR requirement criteria to be used by the SQA organizational element:

a) The methods used to validate that the software product matches the user documentation.
b) Test plans, test procedures, and test cases to ensure that all user documentation (e.g., users and operators manuals) are tested in conjunction with the software.
c) The user documentation should be reviewed for compliance with the approved project documentation standard.

The UDR can be held independently or in conjunction with other reviews.

The results of the review should be documented in a UDR report, which identifies all deficiencies discovered during the review and which provides a plan and schedule for corrective action. The updated user documentation should be placed under configuration management prior to the physical audit described in 3.6.2.6.

3.6.3.2 Quality assurance audit

An audit of the SQAP and procedures may be required by management to ensure its adequacy. This could occur whether the SQA organizational element is independent or part of the development group. It would most likely occur if the SQA organizational element were in the same reporting chain as the development group. The SQAP should indicate the independent organizational element(s) responsible for conducting the QA review and for any corrective action.[14]

3.7 Test

3.7 Test (Section 7 of the SQAP). This section shall identify all the tests not included in the SVVP for the software covered by the SQAP and shall state the methods to be used.[a]

[a]See IEEE Std 829-1983 (Reaff 1991) and IEEE Std 1008-1987 (Reaff 1993).

The SQAP shall include the specific software tests and testing not addressed in the SVVP (Section 4.2.3 of the SQAP) or other test documentation. The SQAP should identify and describe the methods, approaches, and techniques to be used (or reference appropriate test documentation that has been developed for the project). Test methods and techniques can be divided into two classifications: static tests and dynamic tests. Static testing evaluates or analyzes the software without executing the code (e.g., desk checking, code walk-throughs). This section addresses only dynamic testing.

Dynamic testing, the execution of software on any computer, encompasses such methods and techniques as unit level testing, white-box testing, black-box testing, integration testing, system level testing, stress testing, security testing, and acceptance testing. IEEE Std 1008-1987 (Reaff 1993) defines a systematic and documented approach to unit testing.

This section of the SQAP should include a description of the test planning, test execution, and evaluation of the results of testing, or include a reference to the appropriate section of the SQAP or to standards and procedures that describe the testing process. The testing process described should adequately address the preparation and review of documentation associated with the testing process, including test plans, test design and test case specifications, test procedures and test instructions, test schedule, test reports, and test incidents and their resolution.

Test documentation not specifically referenced in Section 7.0 of the SQAP should be delineated, or a reference to the appropriate section of the SQAP or to test documentation standards, procedures, or conventions that describe the test documentation to be provided, should be included in this section. IEEE Std 829-1983 (Reaff 1991) describes a set of basic test documentation associated with dynamic testing.

This section should include a description of the management and organizational responsibilities associated with the implementation of the tests described in this section, or a reference to the appropriate section of the SQAP or standards, procedures, or conventions that describe the management and organizational responsibilities.

This section should include a description of the computer-aided software engineering (CASE) tools used in the support of test activities or a reference to the appropriate section of the SQAP.

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the test plans, test data, and test activities.

[14]See ISO 9000-1: 1994.

3.8 Problem reporting and corrective action

3.8 Problem Reporting and Corrective Action (Section 8 of the SQAP). This section shall:
(1) Describe the practices and procedures to be followed for reporting, tracking, and resolving problems identified in both software items and the software development and maintenance process.
(2) State the specific organizational responsibilities concerned with their implementation.

Problems encountered during software development or operation may result from defects in the software, supporting and development processes, hardware, or operations. Because of their diversity, the determination of the sources of a problem and the appropriate corrective action requires a centrally controlled system for monitoring problems and determining root causes.

The purposes of problem reporting and corrective action systems are to

a) Assure that problems are documented, corrected, and used for process improvement.
b) Assure that problem reports are assessed for their validity.
c) Assume reported problems and their associated corrective actions are implemented in accordance with customer approved solutions.
d) Provide feedback to the developer and the user of problem status.
e) Provide data for measuring and predicting software quality and reliability.

These goals should be satisfied by the problem reporting and corrective action system described in the SQAP.

The SQAP should include methods to be used to assure the reported problems are being properly addressed. The SQAP should describe the organizational element(s), provisions, and timely procedures for documenting, validating, tracking, and reporting the status of problems and the appropriate corrective action.

Validating, tracking, and resolving problems require the coordination of various groups within the organization. The SQAP should specify the groups responsible for authorizing and implementing problem reporting and corrective actions, and submission of unresolved issues to management for resolution. The problem reporting and corrective action system is usually implemented by SCM and documented in the SCMP. Also, it should identify the point in the development process where generation of problem reports is to begin for each class of development result (e.g., plans, SRS, SDD, code, test documentation).

The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the use of problem reporting and corrective action practices and procedures.

3.9 Tools, techniques, and methodologies

3.9 Tools, Techniques, and Methodologies (Section 9 of the SQAP). This section shall identify the special software tools, techniques, and methodologies that support SQA, state their purposes, and describe their use.

The SQAP shall identify the tools, techniques, and methodologies to be used to support software quality assurance. It should list or reference those tools, techniques, and methodologies that are available, and those that need to be acquired or developed. The responsible organization(s) also should be identified.

3.9.1 Tools

SQA software tools aid in the evaluation or improvement of software quality. Typical tools include, but are not limited to, operating system utilities, debugging aids, documentation aids, checklists, structuring preprocessors, file comparators, structure analyzers, code analyzers, standards auditors, simulators, execution analyzers, performance monitors, statistical analysis packages, software development folders/files, software traceability matrices, test drivers, test case generators, static or dynamic test tools, and information engineering CASE tools.[15]

3.9.2 Techniques

SQA techniques are technical and managerial procedures that aid in the evaluation and improvement of software quality. Such techniques include review of the use of standards, software inspections, requirements tracing, requirements and design verification, reliability measurements and assessments, and rigorous or formal logic analysis.

3.9.3 Methodologies

SQA methodologies are integrated sets of the above tools and techniques. The methodologies should be well-documented for accomplishing the task or activity and provide a description of the process to be used. For example, if the SQA staff uses an RVTM as a method to verify and validate the requirements, design, and test phases, a sample should be provided in the SQAP. It should describe how and when (at what control gates during the development cycle) it will be used.

3.10 Code control

3.10 Code Control (Section 10 of the SQAP). This section shall define the methods and facilities used to maintain, store, secure and document controlled versions of the identified software during all phases of the software life cycle. This may be implemented in conjunction with a computer program library. This may be provided as a part of the SCMP. If so, an appropriate reference shall be made thereto.

Code control can be interpreted as the ways and means necessary to protect or ensure the validity of completed code. Once an appropriate baseline has been established, the code should be placed under configuration management in a computer program library. The SQAP should specify controls (or reference the CM procedures for software change) and security measures for software change and for protection from inadvertent alteration after the code has been baselined. It should define or reference the CM procedures (e.g., see SCMP or the CM section in the SDP) and organizational responsibility for controlling the developed code.

The SQAP should specify or reference a code control procedure that

a) Defines the specific software to be controlled.
b) Describes a standard method for identifying, labeling, and cataloging the software.
c) Lists the physical location of the software under control.
d) Describes the location, maintenance, and use of all backup copies.
e) Describes procedures for distributing a copy.
f) Identifies the documentation that is affected by changes.
g) Describes procedures for implementing a new version.
h) Describes how compliance with the above is assured.

[15]See IEEE Std 1209-1992.

3.11 Media control

3.11 Media Control (Section 11 of the SQAP). This section shall state the methods and facilities to be used to (a) identify the media for each computer product and the documentation required to store the media, including the copy and restore process, and (b) protect computer program physical media from unauthorized access or inadvertent damage or degradation during all phases of the software life cycle. This may be provided as a part of the SCMP. If so, an appropriate reference shall be made thereto.

Computer program media can be defined as those media on which computer data are stored. Typically, the storage media are CD-ROM, RAM, disks, or tapes, but could include cards, diskettes, listings, or other forms on which the data reside.

The media control methods and facilities should ensure that

a) The software is stored and retrieval is assured.
b) Off-site storage and retrieval are provided for critical software and copies of baselined code.
c) The software is accessible only to those with the need of access.
d) The environment is controlled so that the physical media on which the software is stored do not degrade.
e) A description is provided of how compliance with the above is assured.

The SQAP should reference (e.g., see SCMP or CM section of the SDP) or specify procedures and practices that pertain to the above items. For example, a backup procedure for software could indicate the schedule for backup, the type of media on which it will be placed, the location of the storage, the environment of the storage area, and the method to retrieve the backed-up software. A security system may be in place that allows access to software only through an authorization process. The Software Security Plan (3.4.4.9) should be referenced. The SQAP should delineate the organization elements responsible for administering and reviewing media control methods and facilities. The method for identifying, labeling, and data logging may be the same in both code and media control.

3.11.1 Unauthorized access

Several methods are available that will provide adequate protection from unauthorized access of computer program media. The primary method is to provide a permanent labeling or identification scheme within the storage media. When a disk or tape is used on a computer, this technique can provide adequate password control or access protection. Other methods include a limited access program library, encryption, external markings, and proprietary statements identifying a controlled program. The physical security of all media also must be considered.

SQA activities to verify appropriateness and implementation of access procedures should be documented in the SQAP. Areas of concern include identifying the programs requiring limited access, adherence to label and file restrictions, ensuring use of adequate external labeling restrictions, and providing a controlled environment such as a program library.

3.11.2 Inadvertent damage or degradation

Damage or degradation of the media can be minimized by providing adequate configuration management techniques, safe storage locations such as fireproof vaults, and packaging practices that are antistatic in design. Periodic review to ensure use of controlled environmental and cataloging practices will minimize degradation of the media.

SQA activities to verify appropriateness and implementation of procedures to minimize media damage or degradation should be documented in the SQAP.

3.12 Supplier control

3.12 Supplier Control (Section 12 of the SQAP). This section shall state the provisions for assuring that software provided by suppliers meets established requirements. In addition, this section shall state the methods that will be used to assure that the software supplier receives adequate and complete requirements. For previously-developed software, this section shall state the methods to be used to assure the suitability of the product for use with the software items covered by the SQAP. For software that is to be developed, the supplier shall be required to prepare and implement a SQAP in accordance with this standard. This section shall also state the methods to be employed to assure that the developers comply with the requirements of this standard.

This section of the purchaser's SQAP should specify

a) The purchaser's involvement with the supplier's SQA program.
b) The purchaser's procedures for auditing the supplier's conformance to IEEE Std 730-1989, or equivalent contractually standard, and the supplier's SQAP (an option could be to provide for an independent auditor).
c) The actions available to the purchaser should the supplier not be in conformance with IEEE Std 730-1989 and the supplier's SQAP.

3.13 Records collection, maintenance, and retention

3.13 Records Collection, Maintenance, and Retention (Section 13 of the SQAP). This section shall identify the SQA documentation to be retained, shall state the methods and facilities to be used to assemble, safeguard, and maintain this documentation, and shall designate the retention period.

SQA records collection, maintenance, and retention procedures in the SQAP should conform to the approved SCMP methods. To avoid redundancy, a reference to the SCMP procedures could be noted in the SQAP.

3.13.1 Records collection

The type of records to be collected, whether softcopy or hardcopy, are determined by the overall objectives for record keeping. These objectives should be documented in the SQAP. Possible objectives are

a) To provide contractual evidence that the software development process was performed in conformance with established practice and the customer's requirements:
   1) The SQAP is being followed and conforms to the requirements of applicable standards.
   2) The software meets design intent and satisfies contractual requirements.
   3) Corrective action is effective, timely, and complete (i.e., action items are tracked until resolution).
   4) Testing has been performed in accordance with test plans.
b) To provide historical or reference data that could be used to discover long-term trends in the organization's development techniques. The documents collected for historical or reference purposes should be capable of providing data for productivity, quality, and methodology studies. Metrics collected should be reviewed for trends and process improvement. The documents should provide sufficient design, implementation, and testing data so as to be useful for future development.

In addition to SQA documents, records should include program media containing the exact version of programs and materials used in performing tests to assure, test repeatability at any time in the future.

3.13.2 Records maintenance

The SQAP should specify the manner in which records will be kept, that is, softcopy, hardcopy, microfiche, etc. Also, it should state how records will be stored to protect them from fire, theft, or environmental deterioration. The SQAP should provide for historical archiving if applicable.

3.13.3 Records retention

The SQAP should specify the length of retention for each type of record maintained. It is important to state in the SQAP when records should be retained and when they should be destroyed.

3.14 Training

3.14 Training (Section 14 of the SQAP). This section shall identify the training activities necessary to meet the needs of the SQAP.

The need for training of personnel designated to perform the activities defined in the SQAP should be assessed. It is suggested that a matrix be created that defines the task, skill requirement(s) of the task, and the skills of the personnel designated to perform the task. This will allow the rapid identification of training needs for each task and each individual. Considerations for training should include special tools, techniques, methodologies, and computer resources. The SQA organizational element's tasks should include how to evaluate and report quality of the following:

a) Documentation (e.g., SCMP)
b) Standards, practices, conventions, and metrics
c) Reviews and audits
d) Testing
e) Problem reporting and corrective action
f) Code, media, and supplier control
g) Records collection, maintenance, and retention
h) SQAP evaluation
i) SQA management

Once the training need is established, a training plan should be developed that identifies the training activities required to successfully implement the SQAP. Existing training programs should be adapted or new training programs developed to meet these training needs. Training sessions should be scheduled for personnel who will be assigned to carry out the tasks. This training should be compatible with the task schedules discussed in 4.3.2.

If training for non-SQAP related software activities is not covered in other project documentation, the SQA organizational element, upon completion of the other project documentation review, should recommend the development and inclusion of a training matrix into their documentation. This matrix should be used by the SQA organizational element to evaluate all project training. The matrix should be updated (person and course) as training is successfully completed.

3.15 Risk management

3.15 Risk Management (Section 15 of the SQAP). This section shall specify the methods and procedures employed to identify, assess, monitor, and control areas of risk arising during the portion of the software life cycle covered by the SQAP.

Each risk area, whether technical, economic, schedule, managerial, marketing, or any other kind, shall be identified. The risk identification shall be justified. For example: a) Why does the SQA staff believe it to be an area of risk? b) With whom has the SQA staff consulted (identify by job title not name)? c) What information has reached the SQA staff leading them to believe this is a risk area? The risk factors (risk composition) and acceptable results of the risk should be identified. It is essential to identify the level of risk (high, medium, or low) associated with the possible failure of each required task or the unavailability of each resource. The risk assessment should identify the resulting impact on schedules and outline alternative actions available to mitigate the risks.

The risk assessment should quantify the magnitude of each identified risk. Managers should monitor risks in their assigned areas and present periodic reports on their status. Managers also should develop plans for the elimination or reduction of the risks in their assigned areas and report on their progress in implementing these risk control plans. A separate risk management plan may be appropriate for projects that have been identified as having potential high risks. Each risk assessment item should be documented under the headings: a) Description, b) Impact, c) Monitoring, and d) Mitigation.

As the risks change throughout the life cycle, the SQAP should be updated (e.g., a reprioritization of resources, additional prototyping, walkthroughs, testing, or reviews should be conducted).

4. Implementation of a software quality assurance plan

The purpose of this clause is to describe the steps necessary for successfully implementing the SQAP that has been prepared for a specific project. The following items are discussed in this clause:

a) Acceptance of the SQAP by the software developers and others whose task responsibilities are defined in the SQAP.
b) Acceptance of the SQAP by management.
c) Planning and scheduling of resources and tasks for implementation of the SQAP.
d) Distribution of the SQAP to implementors and interfacing personnel.
e) Execution of the SQAP.

4.1 Acceptance by development and other personnel

It is essential to foster a spirit of cooperation between the personnel responsible for software development and the SQA activities. An effective method of achieving this is to have all concerned personnel participate in the preparation of the SQAP. This will tend to increase their support of SQA in general, and of the SQAP in particular. Preliminary drafts of the SQAP should be circulated within the development organization for review and comments. Also, it may be useful to hold walkthroughs of the SQAP with all concerned personnel. During this time they will be able to ask questions directly of the authors of the SQAP, and to make their concerns and objections known before the SQAP is officially published. In this manner, the groundwork will be laid for cooperation and mutual support between all organizations responsible for activities required by the SQAP.

4.2 Acceptance by management

Management acceptance and commitment to the SQAP should be obtained. This will provide the support required for implementing the tasks defined. This acceptance should include commitments for the budget and resources required to implement to the SQA activities.

The SQAP should be coordinated with, and agreed to, by each unit of the organization having responsibilities defined within the SQAP. Acceptance of the SQAP should be indicated on the cover page (hardcopy or softcopy) by an approval signature of the person in charge of each unit. Implementation of the SQAP can be effective only if all the actions called for in the SQAP are performed with the full support of management.

4.3 Planning for implementation of the SQAP

Planning for SQAP implementation comprises two aspects:

a) Identification of required resources.
b) Scheduling implementation resources.

4.3.1 Resources

The four types of resources required to implement a SQAP are personnel, equipment, facilities, and tools. The quantity and quality of these resources should be made known to the appropriate level of management.

The responsible element should identify the job classifications and skill levels of the personnel required to implement and maintain the SQAP throughout the life of the project. It should identify the hardware needed to implement the SQAP and to support it throughout the project, as well as estimates of computer time and support required. Also, it should identify the facilities needed for storage of media and records. When resources are identified by an element other than the SQA element, the SQA element should verify compliance with this task. The tools required for implementation should have been already identified in the SQAP itself.

4.3.2 Scheduling

Once the resources involved in implementing a SQAP have been identified, the next step is to establish a schedule for the implementation. The SQA schedule should be established in consonance with the development schedule. For each task identified in SQAP, this schedule should identify the starting and completion dates for each required resource. In a similar manner, a schedule should be established for the development or acquisition of any necessary support tools.

The schedule information should be coordinated with the development team and manager. The program manager, development manager, and the QA manager should be using the same schedule for the total program. A copy of the coordinated, approved schedule could be incorporated into this clause as well as in the SDP and/or the SPMP. The SPMP should contain an overall schedule for all measurable milestones.

4.4 Distribution of the SQAP

The following applies whether the SQAP is distributed by softcopy or hardcopy. A distribution list of all personnel who are to receive the final approved copy of the SQAP should be prepared. A copy of the published SQAP should then be distributed (physically or electronically) to each individual listed, with an attached sign-off sheet that is to be initialed by the person receiving the SQAP and returned to the organization responsible for the SQAP's publication and distribution.

Follow-up should be done if the sign-off sheet is incomplete or not returned. This helps to ensure intended recipients have the correct version of the SQAP.

4.5 Execution of the SQAP

Once the SQAP is distributed, the SQA organizational element, or other designated organization, should assure that the tasks (e.g., reviews and audits) documented within the SQAP are performed at the appropriate points in the life cycle. The SQAP should identify the activity to be performed, the organizational element performing the activity, and the results to be achieved. It should reference other documents as necessary. Associated work papers of the reviews and audits should provide sufficient evidence that the steps in the SQAP have been performed and reviewed by the management accountable for the SQAP. This should permit an objective determination of how well the SQA objectives have been met.

5. Evaluation of a software quality assurance plan

5.1 Purpose

The SQA element should make provision for periodic or ongoing evaluation of the SQAP. Evaluating the SQAP involves examining it from two different viewpoints:

a) Evaluating the content of the SQAP (initially and after all revisions).
b) Evaluating the use and management of the SQAP.

The evaluation of the SQAP's content is an assessment of how the SQAP complies with IEEE Std 730-1989, internal development and quality assurance standards, and contractual documents. Evaluation of the completeness and applicability of the SQAP is facilitated by the questions presented in 5.2.1.

These questions provide an overview of the state of the SQAP.

The evaluation of the use and management of the SQAP is an assessment of the specific project's implementation of the SQAP. Some suggestions for this ongoing activity are contained in 5.2.2.

5.2 Methodology

The methodologies used to evaluate the SQAP should include reviewing the documentation and observing the implementation process. A checklist of questions should be developed in accordance with the documentation standards approved for project use.

5.2.1 SQAP evaluation

The following questions should be asked in evaluating the overall approach of the SQAP:

a) Are all mandatory requirements of IEEE Std 730-1989 addressed in the SQAP?
b) Are all contractual and company SQAP standards addressed in the SQAP?
c) Does the SQAP specify compliance with any standards in addition to IEEE Std 730-1989? If so, does the SQAP meet the requirements of those standards?
d) Are all exceptions to mandatory requirements noted and adequately justified?
e) Is the content of the SQAP adequate to achieve its stated objectives?
f) Are the SQA requirements, as stated, enforceable, and measurable?

Additional questions that can be used in support of the evaluation of specific SQAP sections are

g) Purpose
   1) Are the specific purpose and scope of the SQAP described?
   2) Are the software product items covered by the SQAP completely described?
   3) Is the intended use of the software items described?

h) Referenced documents
   1) Are all documents referenced by the SQAP listed in this section?

i) Management
   1) Are the structures of all the organizations that influence the quality of the software depicted?
   2) Are the management activities completely described?
   3) Are the tasks and responsibilities of the organizations that influence the quality of the software listed?

j) Documentation
   1) Does this section describe all necessary software documentation?
   2) Does this section describe the methodologies to be used for checking documentation adequacy with reference to 3.6, Reviews and Audits?
   3) Are the methodologies adequate?

k) Standards, practices, conventions, and metrics
   1) Does this section identify all standards, practices, conventions, and metrics to be applied to the software?
   2) Are compliance and monitoring procedures identified?
   3) Are 1) and 2) adequate?

l) Reviews and audits
   1) Does this section define all necessary reviews and audits for the documentation described in 3.4, Documentation?
   2) Are the methodologies for all reviews and audits described?
   3) Are 1) and 2) adequate?

m) Test
   1) Does this section define additional tests not included in the SVVP?
   2) Are the test methods and techniques described for static and dynamic testing as well as associated QA activities?
   3) Does the test documentation conform to IEEE Std 829-1983 (Reaff 1991)?

n) Problem reporting corrective action
   1) Does this section describe problem reporting and corrective action procedures to be used for this project?
   2) Does this section state specific organizational responsibilities?
   3) Are the procedures adequate?

o) Tools, techniques, and methodologies
   1) Are all tools, techniques, and methodologies to be used for SQA purposes fully described?
   2) Are they adequate?

p) Code control
   1) Does this section contain a description of all methods and facilities to be used for code control?
   2) Are they adequate?
   3) Does this section describe how compliance will be achieved?

q) Media control
   1) Does this section contain a description of all methods and facilities to be used for media control?
   2) Are they adequate?
   3) Does this section describe how compliance will be achieved?

r) Supplier control
   1) Are all procedures for interfacing between each supplier’s SQAP and this SQAP fully described?
   2) Are they adequate?

s) Records collection, maintenance, and retention
   1) Are all records collection, maintenance, and retention procedures fully described?
   2) Are they adequate?

t) Training

1) Does this section identify the training activities associated with the requirements of this SQAP?

2) Is the training plan complete (i.e., does it address all of the training activities associated with this SQAP)?

u) Risk management

1) Does this section identify the level of risk (high, medium, or low) associated with each task or resource required?

2) Are the methods and procedures which identify, assess, monitor, and control areas of risk described in this SQAP?

3) Are the identified risks mitigated?

5.2.2 SQAP implementation evaluation

At several points in the product life cycle, usually major project milestones, the SQAP and its implementation should be evaluated by means of a managerial review. This will help assure that the project and its SQAP evolve together. As the project proceeds through the software life cycle, there are likely to be changes in the product scope. As the development plan changes, the SQAP and its implementation also should be reviewed to determine if any changes are required.

The use of the SQAP should be evaluated in terms of the tasks and responsibilities detailed in the SQAP (3.3.2 and 3.3.3). This evaluation should review the status of each task and the adequacy of the actions taken in terms of both product quality results and the schedules actually achieved for the tasks.

5.2.3 SQAP evaluation process relationship

The evaluation process will have a cause-effect relationship as shown in figure 1. An SQAP evaluation, whether formal or informal, may have the effect of causing SQAP modification or causing an implementation evaluation. An SQAP modification may necessitate a corresponding implementation modification. An implementation evaluation may cause an implementation change to bring the use and management of the SQAP into compliance with the SQAP.

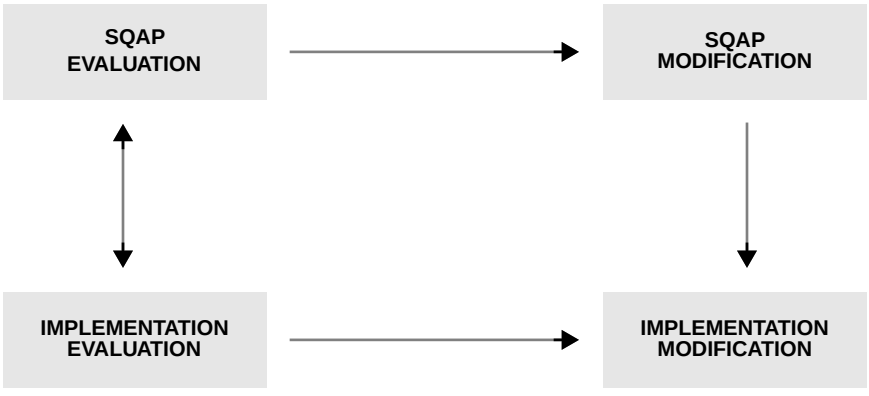

Figure 1—Cause-Effect of SQAP evaluation and modification

6. Modification of a software quality assurance plan

The previous clause addressed the evaluation of an SQAP and the determination of any necessary changes to it. This clause will describe a mechanism for implementing such changes.

6.1 Purpose

The purpose of this subclause is to provide a method for modifying an existing SQAP. Only if there is a provision for systematic modification of an SQAP can its users have confidence in its continued usability.

There are several reasons why an SQAP, once approved and implemented, may subsequently need to be modified. First, the SQAP may contain deficiencies. Second, it may be necessary to adjust to changes in the environment of the SQAP. For example, a new set of system requirements may require more stringent or more detailed testing to assure they are satisfied. Third, changes in the management structure of the project may make portions of the SQAP (e.g., reporting lines of sign-off authorities) obsolete. Finally, the advent of new processes and technology may make modification desirable, as, for example, when new SQAP tools or techniques must be incorporated.

6.2 Scope

This subclause addresses methods for proposing, reviewing, and instituting modifications to an SQAP. It does not cover modifications to the manner in which the SQAP is used, managed, or controlled; provisions for these are made either within the SQAP itself or in project management directives.

6.3 Methodology

There are five steps in the modification of an SQAP:

a) Identify alternative options.
b) Recommend proposed change.
c) Review proposed change.
d) Incorporate approved change.
e) Release and promulgate change.

Steps a) and b) should be followed in all cases. If a project SCM organization exists, then steps c), d), and e) should be accomplished according to that organization’s procedures. If there is no project SCM organization, then steps c), d), and e) should be followed as described in the subclauses below.

6.3.1 Identify alternative options

Changes to an SQAP may be proposed from any of several sources, such as project management, software development, system validation, configuration management, quality assurance, or the customer. They could suggest different solutions to the same problem. It is important to provide for the results of the SQAP evaluation process (see clause 5) to be routed through all of these sources in order that each of their proposed solutions may be presented and reviewed.

6.3.2 Recommend proposed change

A Configuration Control Board (CCB) should be organized to review all alternative solutions and to determine a single recommended change (or set of changes) that they believe best addresses the acknowledged requirement. Depending upon the frequency with which SQAP changes are proposed, this CCB may be either a standing or an ad hoc organization. It may be useful to set up such a group as a standing organization

at the time the SQAP is first published if numerous change requests are expected. When such requests become fewer and farther between, the CCB may be converted to an ad hoc status.

6.3.3 Review proposed change

Once the CCB has agreed upon a proposed change, it should be sent to all interested or potentially affected parties for their review and comments. This step is necessary to provide agreement before the change is published and distributed. The CCB should have responsibility for evaluation and incorporation of comments received from the reviewers, and for approval or rejection of the proposed change.

6.3.4 Incorporate approved change

If, after studying the reviewers' comments, the CCB approves the proposed change, it is incorporated into the SQAP. Standard document control procedures should be employed here, including editorial review, printing of change pages, use of vertical bars to highlight added or modified text, and control procedures to preserve previous versions of the document.

6.3.5 Release and promulgate change

A management official should be designated who will have sign-off authority on SQAP changes.

Once this official has approved the change page(s) for release, standard document distribution methods may be employed. Then, all that remains is to monitor the implementation of the change. This responsibility should be assigned to the appropriate management official. At this point, the evaluation process begins again (see clause 5).

7. Bibliography

These documents, although not specifically cited in the text of this guide, are related and are provided for information.

ASME NQA-1-1994, Quality Assurance Program Requirements for Nuclear Facilities.[16]

IEEE Std 7-4.3.2-1993, IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations (ANSI).

IEEE Std 603-1991, IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations (ANSI).

IEEE Std 1002-1987 (Reaff 1992), IEEE Standard Taxonomy for Software Engineering Standards (ANSI).

[16]ASME publications are available from the American Society of Mechanical Engineers, 22 Law Drive, Fairfield, NJ 07007, USA.

Annex A

(informative)

Summary of the SQAP and related standards

Table A.1—Summary of the SQAP and related standards

Item	Shall	Should	May	IEEE Std 730-1989	IEEE Std 730.1-1995	Other standards[a]
—Description of specific scope and purpose of SQAP	X			3.1	3.1	
• Intended use	X			3.1	3.1a)	
• Scope	X				3.1b)	
• Reason for SQAP		X			3.1c)	
• Products covered by the SQAP	X			3.1	3.1d)	
• Software life cycle/products		X			3.1e)	IEEE Std 1074-1995
• Base documents		X			3.1f)	
• Rationale for departures from base documents		X			3.1g)	
—Reference documents list		X		3.2	3.2	
—Description of project and plan management	X			3.3	3.3	
• Organization	X			3.3.1	3.3.1	
• Tasks	X			3.3.2	3.3.2	IEEE Std 1074-1995
• Responsibilities	X			3.3.3	3.3.3	
—Identification of documents to be used for development, verification, use and maintenance of the products covered by SQAP and how they are to be evaluated	X			3.4	3.4	
• SRS	X			3.4.2.1	3.4.2.1	IEEE Std 830-1993
• SDD	X			3.4.2.2	3.4.2.2	IEEE Std 1016-1987
• SVVP	X			3.4.2.3	3.4.2.3	IEEE Std 829-1983, IEEE Std 1008-1987, IEEE Std 1012-1986
• SVVR	X			3.4.2.4	3.4.2.4	
• User documentation	X			3.4.2.5	3.4.2.5	IEEE Std 1063-1987

Table A.1—Summary of the SQAP and related standards *(continued)*

Item	Shall	Should	May	IEEE Std 730-1989	IEEE Std 730.1-1995	Other standards[a]
• SCMP	X			3.4.2.6	3.4.2.6	IEEE Std 828-1990, IEEE Std 1042-1987, IEEE Std 1033-1985
• SDP			X	3.4.3(1)	3.4.3.1	
• SPM			X	3.4.3(2)	3.4.3.2	
• SPMP			X	3.4.3(3)	3.4.3.3	IEEE Std 1058.1-1987
• SMM			X	3.4.3(4)	3.4.3.4	IEEE Std 1219-1992
• User requirements statements			X	—	3.4.4.1	
• External interface specifications			X	—	3.4.4.2	
• Internal interface specifications			X	—	3.4.4.3	
• Operations manual			X	—	3.4.4.4	
• Installation manual			X	—	3.4.4.5	
• Training manual			X	—	3.4.4.6	
• Training plan			X	—	3.4.4.7	
• Software metrics plan			X	—	3.4.4.8	IEEE Std 982.1-1988, IEEE Std 982.2-1988
• Software security plan			X	—	3.4.4.9	
—Identification of standards, practices, conventions, and metrics, and statement of compliance check methods	X			3.5	3.5	
• Documentation standards	X			3.5.2(1)	3.5.2	IEEE Std 830-1993
• Logic structure standards	X			3.2.2(2)	3.5.2	
• Coding standards	X			3.5.2(3)	3.5.2	
• Commentary standards	X			3.5.2(4)	3.5.2	
• Testing standards and practices	X			3.5.2(5)	3.5.2	
• Metrics	X			3.5.2(6)	3.5.2	
• Requirements		X		—	3.5.2.1	
• Design		X		—	3.5.2.2	
• Implementation		X		—	3.5.2.3	

Table A.1—Summary of the SQAP and related standards *(continued)*

Item	Shall	Should	May	IEEE Std 730-1989	IEEE Std 730.1-1995	Other standards[a]
• Test		X		—	3.5.2.4	IEEE Std 829-1983, IEEE Std 1008-1987
• Maintenance		X		—	3.5.2.5	IEEE Std 1219-1992
• Documentation		X		—	3.5.2.6	
• Metrics				—	3.5.2.7	IEEE Std 982.1-1988, IEEE Std 982.2-1988, IEEE Std 1045-1992, IEEE Std 1061-1992
—Reviews and audits	X			3.6	3.6	
—Definition of technical review and audits and means of accomplishment	X			3.6.1	3.6.1	IEEE Std 1028-1988, IEEE Std 1074-1995
—Conduct the following reviews:	X			3.6.2	3.6.2	
• SRR	X			3.6.2.1	3.6.2.1	IEEE Std 830-1993
• PDR	X			3.6.2.2	3.6.2.2	
• CDR	X			3.6.2.3	3.6.2.3	
• SVVPR	X			3.6.2.4	3.6.2.4	
• Functional audit	X			3.6.2.5	3.6.2.5	
• Physical audit	X			3.6.2.6	3.6.2.6	
• In-Process audits	X			3.6.2.7	3.6.2.7	
• Managerial reviews	X			3.6.2.8	3.6.2.8	
• SCMPR	X			3.6.2.9	3.6.2.9	
• Post mortem	X			3.6.2.10	3.6.2.10	
• UDR			X	3.6.3	3.6.3.1	
• Quality assurance audit			X	—	3.6.3.2	ISO 9000-1: 1994
—Test	X			3.7	3.7	IEEE Std 829-1995, IEEE Std 1008-1987
—Describe problem reporting and corrective action	X			3.8	3.8	
—Describe tools, techniques and methodologies to be used	X			3.9	3.9	IEEE Std 1209-1992
—Definition of code control methods and facilities	X			3.10	3.10	

Table A.1—Summary of the SQAP and related standards *(continued)*

Item	Shall	Should	May	IEEE Std 730-1989	IEEE Std 730.1-1995	Other standards[a]
—Definition of media control methods and facilities	X			3.11	3.11	
• Unauthorized access	X			3.11(a)	3.11.1	
• Damage and degradation	X			3.11(b)	3.11.2	
—Provision for supplier quality assurance methods	X			3.12	3.12	
—Identification of SQA records collection, maintenance, and retention	X			3.13	3.13	
—Training	X			3.14	3.14	
—Risk management	X			3.15	3.15	
—Implementing an SQAP		X		—	4	
—Evaluating an SQAP		X		—	5	
—Modifying an SQAP		X		—	6	

[a]Note that reaffirmation dates are not included in this table. For information about references, see clause 2 of the standard.