

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

Copyright © 1996 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1996. Printed in the United States of America.

ISBN 1-55937-593-0

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

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.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (508) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

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 Gershtkovitch
 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. Schnoegel
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 Rüschi
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

CLAUSE	PAGE
1. Overview.....	1
1.1 Scope.....	1
1.2 Purpose.....	1
2. References, definitions, and acronyms	2
2.1 Specific references	2
2.2 Definitions.....	3
2.3 Abbreviations and acronyms.....	3
3. Contents of a Software Quality Assurance Plan.....	4
3.1 Purpose.....	5
3.2 Reference documents	5
3.3 Management.....	6
3.4 Documentation.....	8
3.5 Standards, practices, conventions and metrics.....	18
3.6 Reviews and audits	20
3.7 Test.....	30
3.8 Problem reporting and corrective action.....	31
3.9 Tools, techniques, and methodologies	31
3.10 Code control.....	32
3.11 Media control	33
3.12 Supplier control.....	34
3.13 Records collection, maintenance, and retention	34
3.14 Training.....	35
3.15 Risk management.....	36
4. Implementation of a software quality assurance plan.....	36
4.1 Acceptance by development and other personnel.....	36
4.2 Acceptance by management	37
4.3 Planning for implementation of the SQAP	37
4.4 Distribution of the SQAP.....	37
4.5 Execution of the SQAP	38
5. Evaluation of a software quality assurance plan.....	38
5.1 Purpose.....	38
5.2 Methodology	38
6. Modification of a software quality assurance plan	41
6.1 Purpose.....	41
6.2 Scope.....	41
6.3 Methodology	41
7. Bibliography	42
ANNEX	
Annex A (informative) Summary of the SQAP and related standards.....	43

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.

¹Information 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).

²IEEE 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.

³IEEE 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

⁴ISO 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 Report
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.

⁵See 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-

⁶See 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

^aSee 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.

⁷See 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

^aSee 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.