SOFTWARE QUALITY ASSURANCE VOLUME II A Program Guide

James Vincent

Albert Waters

John Sinclair

SOFTWARE QUALITY ASSURANCE VOLUME II A Program Guide

James Vincent

Albert Waters

John Sinclair

PRENTICE HALL, *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging-in-Publication Data
(Revised for volume 2)

VINCENT, JAMES
Software quality assurance.

Bibliography: v. 1, p.
Includes indexes.
Contents: v. 1. Practice and implementation—v. 2.
A program guide.
1. Computer software—Quality control.
I. Waters, Albert II. Sinclair, John III. Title.
QA76.76.Q35V56 1988 005 87-19359
ISBN 0-13-821860-9 (v. 1)
ISBN 0-13-823139-7 (v. 2)

Editorial/production supervision
and interior design: BARBARA MARTTINE
Cover design: WANDA LUBELSKA
Manufacturing buyer: MARIANNE GLORIANDE

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-823139-7

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

PREFACE

This text offers a generic example of a Software Quality Assurance (SQA) Program Plan, presenting an actual application of the policies, practices, standards, and management techniques discussed in our other SQA work, Software Quality Assurance: Practice and Implementation.

In actual practice the program Document is a "write-to" document, a characterization which requires some explanation. Typically, the Management Information Systems Division (MISD) Director, and the Directors of the Computer Systems Management Division, Systems Design and Maintenance Division, and Computer Operations Division meet with the Software Quality Assurance Administrator at the outset of the SQA effort to discuss the policies, practices, standards, and methodologies which the SQA function will employ. Among the outcome of these meetings would be a list of topics for the SQA Administrator to "write to," or address, in producing the SQA Program Document.

In this text we have "written to" those topics which we feel should be addressed in four general areas: 1) an overview of the SQA systems, 2) SQA program requirements, 3) design and development controls, and 4) operation controls: these being the four chapter divisions of the text. Due to the overlapping nature of the first three chapters, there is necessarily some duplication of the material presented from chapter to chapter. The approach to the material, however, and the details offered, are in each case tailored to the special nature of the chapter topic.

In terms of the full organization SQA effort the SQA Program Document serves these purposes:

a. Provide an overview of the entire SQA Program and its place in the organization.
b. Act as a tutorial for all organization members involved in the SQA effort.
c. Define the typical organization software system development life-cycle and illustrate the integration of SQA.
d. Establish more detailed guidelines for the SQA function.
e. Outline more detailed responsibilities of organization elements involved in the SQA effort.
f. Provide details of the nature, methods, and purposes of SQA reviews and audits.
g. Outline responsibilities of other product control disciplines and their relation to the SQA function.
h. Further the organization's commitment to the SQA effort and to product quality.

The relationship of the Program Document to other SQA Program documentation is illustrated in Figure 2–3 in Chapter 2 of this text.

If this text had been an applied Program Document, appendices would have been given referencing such elements as detailed review and audit checklists, program coding standards, and detailed instructions on completing the SQA-related control forms. These appendices would have been developed by the SQA Administrator working in conjunction with the Chief Quality Assurance Circle (CQAC) [See section 2–3, chapter 2]. They would have been tailored to the organization's control requirements and standards, and would accompany the Program Document in a separate volume. For some idea of the contents of these appendices see Appendices B and C in this volume. These reproduce listings of Software Quality Factors, Criteria, and Itemized Requirements developed by McCall, Richards, and Walters for the Rome Air Development Center, and published in the Center's study *Factors in Software Quality Assurance* (RADC-TR-77-369, Volume I). For more detail on the theory and practice of Software Quality Assurance presented here, we refer you to our text *Software Quality Assurance: Practice and Implementation* (Prentice-Hall, 1988).

Finally, we leave you with one last note regarding the appendices here. Appendix A presents an alternative life-cycle to the standard life-cycle presented throughout much of this text: the Contemporary Development Life-Cycle (CDLC). This concept was recently developed by the U.S. Army Information Systems Engineering Command to accelerate the development of database oriented systems, and represents the very latest thinking in life-cycle theory. Although some of the documentation requirements are still vague, and while its applicability is limited to database-oriented systems, it shows great promise for both shortening the development life-cycle and ensuring a high degree of software quality. More information on the CDLC may be found in USAISEC Pamphlet No. 25-CLCD, *Procedures for Contemporary Life-Cycle Development,* available from the Department of the Army, Headquarters, U.S. Army Information Systems Command.

JAMES VINCENT

CONTENTS

1

OVERVIEW

1–1 PHILOSOPHY

Most of us would agree to the prima facie truth that if someone purchases a product with the assurance that it will meet a given need, he or she has a right to expect that product to meet that need. This truth is the driving force behind the concept of **Quality Assurance:** *ensuring that the user is given a product that lives up to its specified needs.*

Beyond this, however, any ADP software developer has a vested interest in the SQA discipline for two reasons:

(1) An organization's reputation rests on the quality of its products (the extent to which they dependably meet user needs),

(2) The morale of an organization's employees depends upon their belief in the quality of their work product, and the pride they take in it. Low morale will in turn lead to poor workmanship, low productivity, and a higher turnover rate.

If either of these factors suffer, due to a lack of organization, concern for quality, the organization itself, and what has become for many companies the bottom line—money—profits will suffer proportionally.

A full program of Software Quality Assurance (SQA) is instrumental in controlling the software development process. Properly developed and implemented, such a program will work to ensure that the software products developed meet their specifications exactly, and beyond this, it will work to ensure that the specifications themselves accurately describe the needs which are to be met. The benefits, as the preced-

ing paragraphs suggest, will be both satisfied customers and users, and more highly motivated employees.

Additionally, an SQA program will have a direct effect on lowering the organization's costs in its software development process. Studies have shown that the cost of correcting "bugs" in a software product increases geometrically as the development process progresses. (See Figure 1-1.) By working to eliminate problems early in the development process, and ensuring that newly arising problems are kept to a minimum and are found and corrected early in each phase of the development life-cycle, SQA can work to substantially reduce the organization's production costs.

The following sections in this chapter discuss the system development life-cycles in detail from the perspective of the SQA function, the achievement of configuration control throughout the development process, the role of testing in the SQA function, and the concept of the reviews, audits, and baselines which are to be employed.

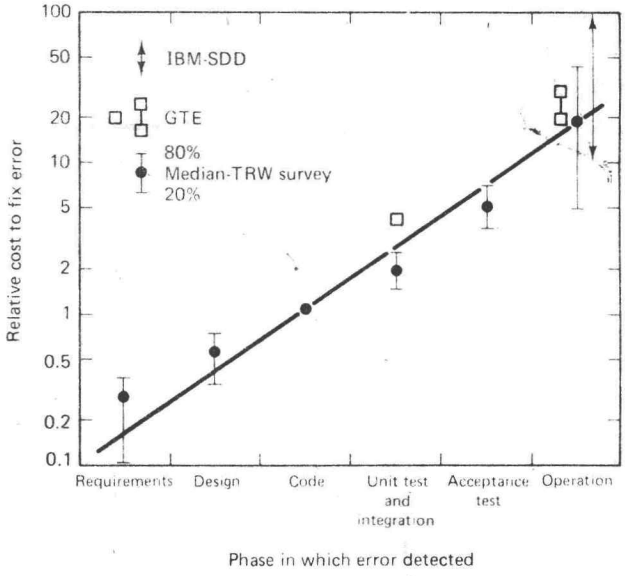


Figure 1-1 Software validation: the price of procrastination.

1-2 PURPOSE

This Program Guide provides guidance and describes the organizational relationships and responsibilities for the development, implementation, and management of a comprehensive SQA Program. The objective of this Program is to establish SQA guidelines and methodology to:

a. Assure compliance of data processing activities with prescribed standards.

b. Provide an environment conducive to an effective Software Quality Assurance system.

c. Address the quality of design and quality of conformance.
d. Provide a Software Quality Assurance program based on a specification system that establishes a direct relationship between performance design requirements and quality assurance provisions.
e. Provide for quality reviews and audits.
f. Provide for the quality assurance of data processing products.

The system and methodology presented in this program document ensure that Software Quality Assurance activities performed throughout the life-cycle of the software development process, and software modification and change activities performed by the software developers, are accomplished within the SQA guidelines, standards, and policies established by the organization.

1-3 APPLICABILITY

The provisions of this document apply to SQA responsibilities for the software development and maintenance activities of the primary software developer, as well as any subordinate components of a particular organization.

1-4 THE SYSTEM LIFE-CYCLES

There are two system life-cycles to be addressed in the development of software products under the control of this SQA Program: the System Development Request Life-cycle (SDRL), and the System Change and Modification Life-cycle (SCAML). Figure 1-2 illustrates the relationship of these two life-cycles. In exceptional cases where maximum control of the software product development is desired, the SCAML may

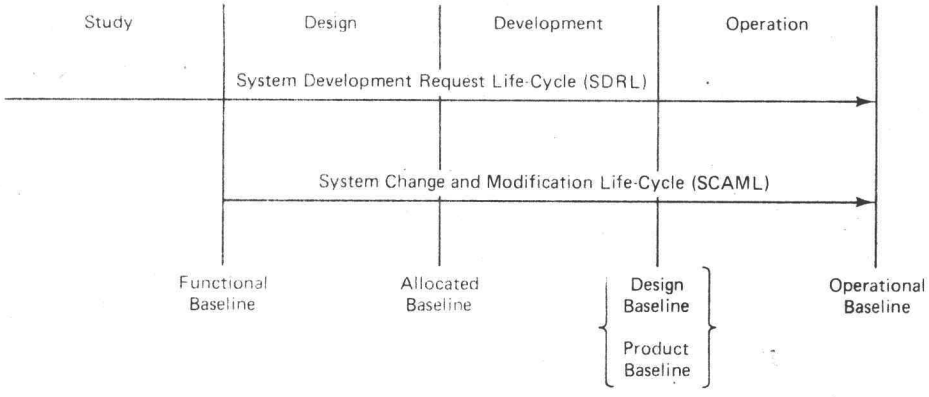


Figure 1-2 Relationship of SDRL to SCAML.

begin after the System Design Requirements Paper (SDRP)* has been drafted and approved, and Computer Program Configurations Items (CPCIs) and Computer Program Components (CPCs) have been identified and approved. Normally, however, as Figure 1-2 suggests, the SCAML will not begin until afer the conclusion of the Study Phase of the SDRL and the sanctioning of the Functional Baseline.

1-5 SYSTEM DEVELOPMENT REQUEST LIFE-CYCLE

Figure 1-3a illustrates the relationship of SQA to the SDRL. The evolution of the SDRL begins with the preparation of a formal System Development Request (SDR) and continues through the subsequent development of more detailed system requirements, the actual coding and testing of the software, and the deployment and operation of the final software product. Steps or tasks toward the accomplishment of SDRL are grouped to form functional, allocated, design, product, and operational baselines from which the configuration of design, development, and the ultimate product can be controlled.

Figure 1-3b offers an alternative illustration of the SDRL, demonstrating the relationship of this life-cycle to both the SQA life-cycle, and to the evolution of project documentation.

The sub-sections which follow present the details of the standard SDRL reviews, including major objectives and documents reviewed. An alternative to the standard life-cycle, the Contemporary Life-Cycle Development (CLCD) approach is offered in Appendix A, together with an overview of the alternative quality reviews.

Initial Design Concept Review and Functional Baseline

The Initial Design Concept Review (IDCR) is accomplished at the end of the Study Phase of the project life-cycle. At the outset of the IDCR the members of the Chief Quality Assurance Circle (CQAC) will review the administrative-generated documentation (Memoranda of Understanding/Agreement, Feasibility Study, Economic Analysis, System Objectives Paper [SOP], and the like) to refresh their understanding of the assumptions and expectations affecting the system to be developed, and to verify them. The system designers will then explain the concerns encountered, the rationale for the system design concept chosen, and the way in which it addresses the requirements outlined in the System Requirements Specification (SRS). With regard to design concerns, this review measures the compliance of the conceptual design with prescribed standards.

Among the primary objectives of the IDCR are these:

1. Verify that a software product (new, or modification of an existing one) is in fact the best approach to the need presented.
2. Verify the economic and technical feasibility of the project.

*A complete listing of the control documents referenced in this text, together with a brief description of their nature and purpose, may be found in the Glossary at the end of the text.

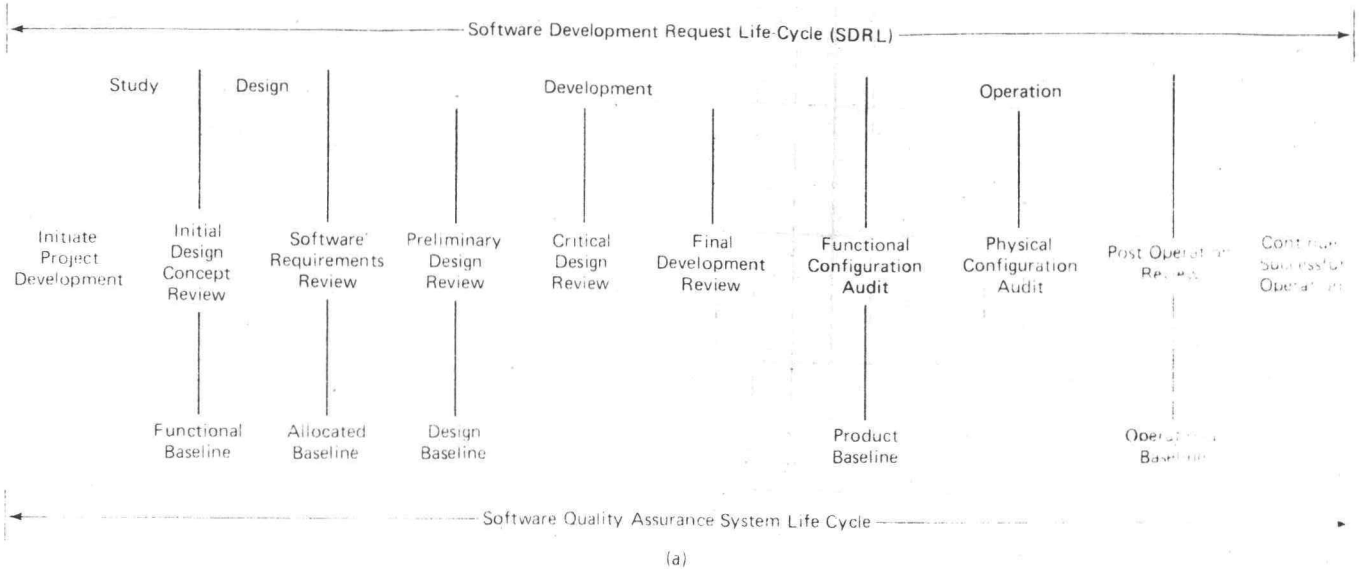
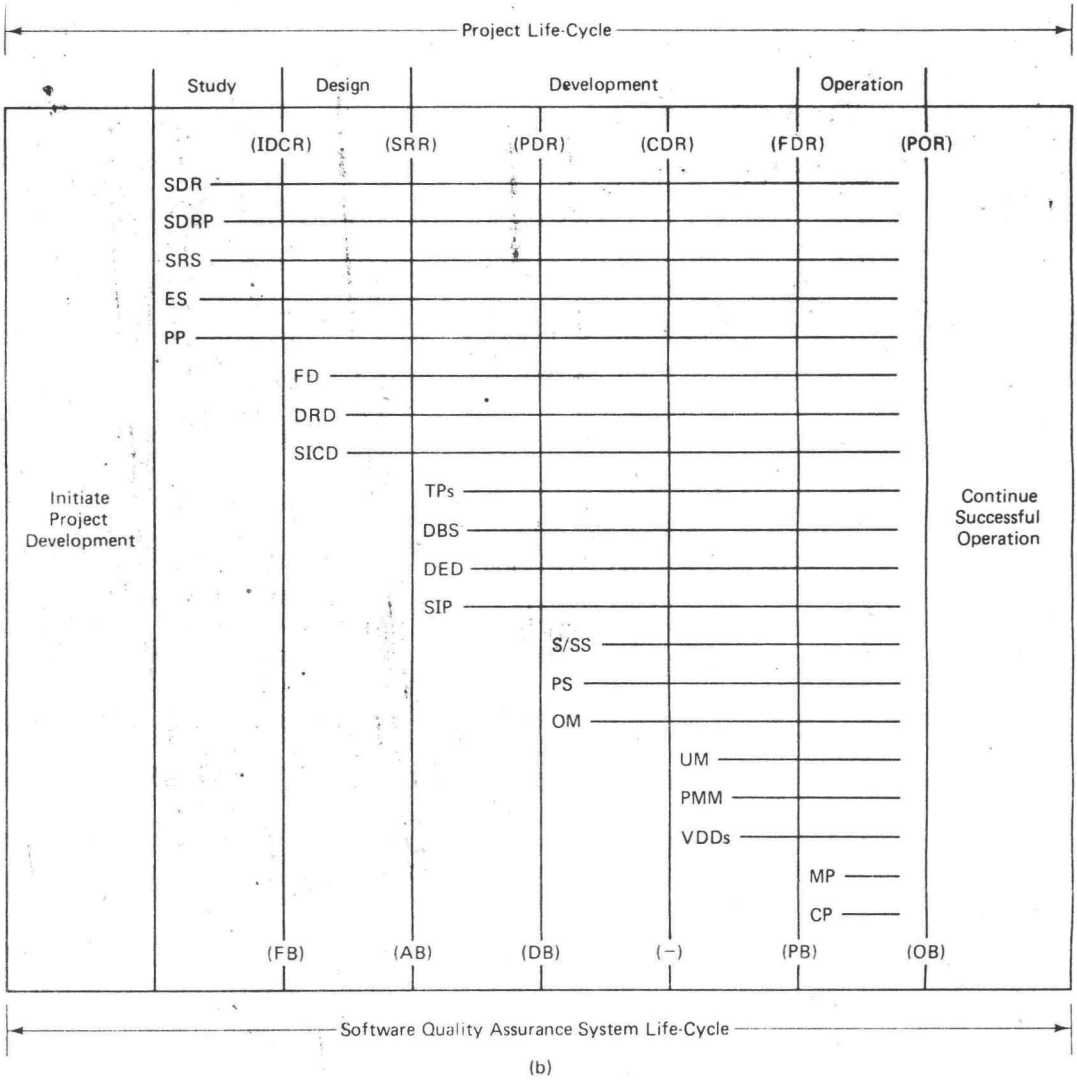


Figure 1-3 (a) Relationship of SQA to SDRL; (b) relationship of project life-cycle to SQA life-cycle and evolution of project documentation.



(b)

Figure 1–3 (cont.)

3. Ensure that the user and the Project Development Team share the same understanding of the system requirements and objectives, and are "speaking the same language."
4. Verify that all documentation is updated to include any new technical information and new requirements presented at this time.
5. Ensure that the initial system requirements are of sufficient depth and detail to permit the initial system design.

6. Ensure that the Software Verification and Validation Plan (SVVP) accurately reflects the developer's understanding of the criticality of the product, and demonstrates that adequate control is planned to ensure product quality.
7. Provide for the initial identification of CPCIs to be tracked for adherence to SQA and other product development controls.
8. Verify the software quality factors required by the product, and the initial identification of the trade-offs and enhancements required.

Documents reviewed should include:

1. Computer Program Configuration Items (CPCIs)
2. [Relevant] Baseline Configuration Item Log (__BCI Log)
3. Configuration Item Index (CII)
4. Discrepancy Report (DR)
5. Discrepancy Report Log (DR Log)
6. Document Review Report (DRR)
7. Economic Analysis (EA)
8. Engineering Change Proposal-Software (ECP-S)
9. Engineering Change Proposal-Software Log (ECP-S Log)
10. Equipment Specification (ES)
11. Feasibility Study (FS)
12. Interim Change Package (ICP)
13. Memoranda of Understanding/Memoranda of Agreement (MOU/MOA)
14. Project Plan (PP)
15. Software Verification and Validation Plan (SVVP)
16. Specification Change Notice(s) (SCN(s))
17. System Decision Paper (SDP)
18. System Design Requirements Paper (SDRP)
19. System Development Request (SDR)
20. System Objectives Paper (SOP)
21. System Problem Reports/System Problem Correction Reports (SPR/SPCRs)
22. System Problem Reports/System Problem Correction Reports Log (SPR/SPCR Log)
23. System Requirements Specification (SRS)*

The IDCR should be conducted by the members of the CQAC seated as a body. Once the Study Phase project documentation is reviewed and approved by the CQAC, the Functional Baseline (FB) is established.

*Appendix D offers an illustration of the documentation "hierarchy" for the SQA control effort, indicating the decomposition, detailing, and documentation of the system requirements, and the levels of control documentation required.

Software Requirements Review

The Software Requirements Review (SRR) is accomplished at the end of the Design phase. During the SRR the system designers will explain the concerns encountered and the rationale used in the development of the Functional Description (FD), Data Requirements Document (DRD), System Interface Control Document (SICD), System/Sub-System Specification (S/SS), Data Elements Dictionary (DED), Data Base Specification (DBS), and any other controlling system documentation produced at this time. With regard to design concerns, this review measures the compliance of the FD, DRD, SICD, S/SS, DED, DBS, and other design documentation with prescribed standards.

Among the primary objectives of the SRR are these:

1. Formalize the organization's commitment to develop the software product.
2. Verify that the initial system design accurately reflects system requirements.
3. Verify that all documentation is updated to include new technical information and new requirements.
4. Ensure that all baseline updates or changes are properly prepared, submitted, approved, and implemented.
5. Ensure that any new CPCIs to be tracked for adherence to SQA and other product development controls are properly identified.

Documentation reviewed should include:

1. Computer Program Configuration Items (CPCIs)
2. [Relevant] Baseline Configuration Item Log (__BCI Log)
3. Configuration Item Index (CII)
4. Data Elements Dictionary (DED)
5. Data Base Specifications (DBS)
6. Data Requirements Document (DRD)
7. Discrepancy Report (DR)
8. Discrepancy Report Log (DR Log)
9. Document Review Report (DRR)
10. Engineering Change Proposal-Software (ECP-S)
11. Engineering Change Proposal-Software Log (ECP-S Log)
12. Equipment Specification (ES)
13. Functional Description (FD)
14. Final Project (or Phase) Report (FPR)
15. Interim Change Package (ICP)
16. Project Plan (PP)
17. Specification Change Notice(s) (SCN(s))

18. System Implementation Plan (SIP)
19. System Interface Control Document (SICD)
20. System Problem Reports/System Problem Correction Reports (SPR/SPCRs)
21. System Problem Reports/System Problem Correction Reports Log (SPR/SPCR Log)
22. System Requirements Specification (SRS)
23. System/Sub-System Specification (S/SS)
24. Test Plans (TPs)

The SRR should be conducted by the members of the CQAC seated as a body. Once the Design Phase project documentation is reviewed and approved by the CQAC, the Allocated Baseline (AB) is established.

Preliminary Design Review and Design Baseline

The Preliminary Design Review (PDR) is accomplished at the end of the first stage of the Development Phase of the life-cycle. During the PDR the system designers will explain the rationale used in the development of the Program Specification (PS), and in resolving detailed program design problems prior to coding. With regard to design concerns, this review measures the compliance of the PS, and any other revised documentation with prescribed standards, and identifies any design problems prior to coding.

Among the primary objectives of the PDR are these:

1. Verify that the final versions of the higher-level system specifications are in accordance with requirements established in the Functional and Allocated Baselines.
2. Verify that the system specifications are of sufficient depth and detail to proceed with detailed design and coding.
3. Ensure that the final CPCIs identified for tracking represent all those which the CQAC wishes to track, and that they are properly identified.
4. Ensure that the proper planning for software testing is taking place [a draft of the Software Development Test (SDT) should be prepared for review].
5. Verify that all documentation is updated to include new technical information and new requirements.
6. Ensure that all initial coding is done in accordance with organization policies, practices, and conventions.
7. Ensure that all baseline updates or changes are properly prepared, submitted, approved, and implemented.
8. Ensure that proper planning for implementation is undertaken [the System Implementation Plan (SIP) should be drafted in this phase].