SANDIA REPORT

SAND85—2344 • UC—32
Unlimited Release
Printed August 1987

RS-8232-2/ 66322
C1
8232-2//066322
00000001 -

Sandia Software Guidelines

Volume 1
Software Quality Planning

DO NOT DESTROY
RETURN TO
LIBRARY VAULT

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

CONTROLLED DOCUMENT
PLEASE ACCEPT IN LDAS

SF2900Q(8-81)

1109144

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

Distribution
Category UC-32

SAND85-2344
Unlimited Release
Printed August 1987

Sandia Software Guidelines

Volume 1

Software Quality Planning

Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

This volume is one in a series of *Sandia Software Guidelines* intended for use in producing quality software within Sandia National Laboratories. In consonance with the IEEE Standard for Software Quality Assurance Plans, this volume identifies procedures to follow in producing a Software Quality Assurance Plan for an organization or a project, and provides an example project SQA plan.

Quality Endorsement

Sandia National Laboratories' stated policy of quality of performance applies to all aspects of our scientific and engineering endeavors. This quest for excellence applies equally to our software pursuits and our hardware products.

The series of documents known as *The Sandia Software Guidelines* represents a consensus by software professionals within the Laboratories to identify and recommend practices that will help produce high-quality software. The Design Engineering Directorate publishes these documents, and the Computing Directorate distributes them. We believe that applying such guidelines reflects good management principles and practices.

As with the IEEE standards which form the basis for these volumes, the *Guidelines* will undergo periodic review for relevancy and current practices. We encourage Sandians to take an active part in that process.

C. Herman Mauney, Director
Systems Evaluation
Sandia National Laboratories

Foreword

This volume is one in a series of Sandia Software Guidelines intended for use in producing quality software within Sandia National Laboratories. These guidelines, when used in conjunction with the IEEE Standard for Software Quality Assurance Plans, will help ensure that computer programs developed within the Laboratories are usable, reliable, understandable, maintainable, and portable. When complete, the series will consist of the following documents:

- Volume 1: **Software Quality Planning** (SAND85-2344)
  Presents an overview of procedures designed to ensure software quality. Includes a sample software quality assurance plan for a generic Sandia project.

- Volume 2: **Documentation** (SAND85-2345)
  Presents a description of documents needed for developing and maintaining software projects. Includes sample document outlines for a generic Sandia software project.

- Volume 3: **Standards, Practices, and Conventions** (SAND85-2346)
  Presents consensus standards and practices for developing and maintaining quality software at Sandia. Includes recommended deliverables for major phases of the software life cycle.

- Volume 4: **Configuration Management** (SAND85-2347)
  Presents a methodology for configuration management of Sandia software projects and their associated documentation.

- Volume 5: **Tools, Techniques, and Methodologies** (SAND85-2348)
  Presents evaluations and a directory of software tools and methodologies available to Sandia personnel.

Acknowledgement

A consensus document like this volume of the guidelines cannot be produced without the cooperation and hard work of a great many people throughout the organization. The sponsoring CAD Technology Division wishes to thank the members of the working group who wrote Volume 1, as well as the members of the balloting group who reviewed and refined it. Special acknowledgement is made to M.J. Adzija and the members of the DOEDEF project, who provided the basis for the sample project quality plan in this document.

Working Group Members, Volume 1

Mike Blackledge, *Chairperson*		Merry Peterson	(2612)
Sandy Babb	(2854)	Jim Rogers	(8474)
Gayle Connor	(9211)	Suzanne Rountree	(2813)
Guy Dahms	(7253)	Bill Shepherd	(6316)
Pete Hamilton	(9241)	Chuck Trauth	(7252)
Chuck Miller	(2813)	John Wisniewski	(2113)
Susan Navarro	(2822)	Ann Yates	(5255)

This document was printed using the LaTeX *computer typesetting program and the Sandia National Laboratories' Autologic APS-5 phototypesetter.*
1 2 3 4 5 6 7 8 9 0

Contents

1 Introduction **1**
- 1.1 Intent 1
- 1.2 Environment 2
- 1.3 Applicability 3
- 1.4 Organization 3
- 1.5 How to Use This Manual 4

2 Software Quality Relationships and Activities **5**
- 2.1 Quality Assurance, Quality Control, and Audits 5
- 2.2 An Overview of Quality Assurance Activities 7
  - 2.2.1 Verification and Validation Activities 8
  - 2.2.2 Configuration Management Activities 8
  - 2.2.3 Quality Control Activities 9
- 2.3 Requirements to Implement an SQA Program 10
- 2.4 Why have an SQA Program? 11

3 Contents of a Software Quality Assurance Plan **13**
- 3.1 Introduction and Program Structure 13
  - 3.1.1 Purpose 13
  - 3.1.2 Reference Documents 14
  - 3.1.3 Management 14
- 3.2 Design Control 15
  - 3.2.1 Design Information 15
  - 3.2.2 Design Review 16
  - 3.2.3 Change Control 17
- 3.3 Other Quality Factors 18
  - 3.3.1 Code Control 18
  - 3.3.2 Media Control 19
  - 3.3.3 Supplier Control 20
- 3.4 Documentation Control 21
  - 3.4.1 Records Collection 22
  - 3.4.2 Records Maintenance 22
  - 3.4.3 Records Retention 23

4 Implementation of a Software Quality Assurance Plan **25**
- 4.1 Acceptance by Management 25
- 4.2 Acceptance by Development Personnel 26
- 4.3 Planning for Implementation of the SQAP 26

4.4 Execution of the SQAP 26

5 Evaluation of a Software Quality Assurance Plan: Guidelines 27

5.1 Evaluation of the Plan's Contents 27

5.2 Evaluation of the Plan's Administration 28

6 Modification of the Software Quality Assurance Plan 30

6.1 Keeping the Plan Current 30

6.2 Reasons for Changing the SQAP 30

6.3 Formality of Change Control 31

6.4 Methodology for Changes 32

Appendix A: Example Software Quality Assurance Plan xxxiv

Appendix B: References 47

Appendix C: Glossary and Acronyms 48

Index 50

List of Tables

1 Example Records for Collection 23

2 Records Retention Periods 24

3 Example SQAP Content Evaluation Form 28

4 Example SQAP Administration Evaluation Form 29

Software Quality Planning

1 Introduction

Software quality is a goal everyone desires; yet not everyone is in agreement regarding

- what constitutes software quality,
- what are the "correct" approaches to achieve it,
- how much effort and time should be spent to achieve it, and
- what are definitive measures or review criteria to determine when it has been achieved.

Management personnel know they want high quality in any software products within their area of control. Software professionals know they want to produce a quality product. Users are adamant that the software "must work right – and be reliable!" All want software quality; but how are they to proceed?

1.1 Intent

As the first volume in a series of *Sandia Software Guidelines*, this manual is designed to help individuals – designers, managers, quality assurance personnel – involved with computer programs and associated documentation at Sandia National Laboratories proceed toward the goal of high quality software. The intent of this manual is to provide guidelines to achieve that quality product. This volume is not intended to provide methods to measure the effect of these or similar guidelines on the software product. The guidelines provided here are customized to the environment found within the Laboratories, as shown in Figure 1.

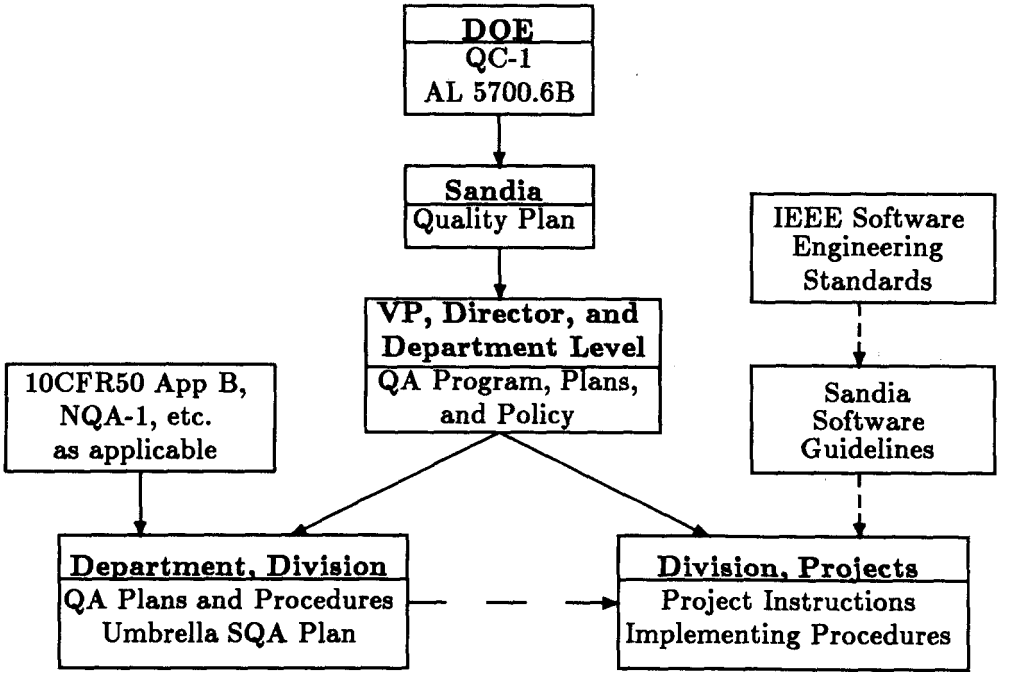

Figure 1: The Sandia Software Quality Environment

1.2 Environment

The Albuquerque Operations Office of the Department of Energy has published two top-level policy documents, known as QC-1 and AL 5700.6B. QC-1 (the *DOE/AL Quality Control Policy*, [QC85]), prescribes the basic principles and requirements for quality assurance programs, and AL 5700.6B (*DOE/AL Quality Assurance*, [AL84]) assigns responsibility for those quality programs.

QC-1 calls for a software quality assurance program at the agency level that establishes strategy, authority, and control. Such a program is to develop verification and validation test plans, change control and configuration control, documentation, and an audit program. In turn, Sandia has published its own *Quality Plan* [SQP86] which further states that subordinate quality plans are to be prepared which are appropriate for the nature of the software or hardware involved. Note that these efforts are aimed at producing organizational (*i.e.*, policy) quality plans, which are usually less detailed than project quality plans.

The general directorate or department level quality plans should specify organizational and policy directives. The details of implementing such plans are left to the division level or project level software quality plans and their implementing procedures. Some organizations within Sandia have already produced detailed Quality Plans for Software Development and Maintenance. As shown in Figure 1, such quality plans may have external requirements, such as 10CFR50 Appendix B, *Quality Assurance Criteria for Nuclear Power Plants and Fuel Reprocessing Plants*, and the expansion given by ANSI/ASME standard NQA-1. The criteria of such documents seldom *all* apply to Sandia. This volume will aid in the preparation of Sandia department and division organizational quality plans in general, and will provide details on producing a Software Quality Assurance Plan for a project within the Laboratories.

1.3 Applicability

This manual is designed for use by any organization or project developing or maintaining software, using either Sandia personnel or personnel under contract to the Laboratories. This document provides guidelines to follow regardless of the application of the software (*e.g.*, WR (War Reserve) or non-WR), the programming language, or the size of the development or maintenance effort.

This manual is a set of *guidelines*, not directives. Enforcement of an application of these guidelines must be established at the organizational level, project level, or individual level, with management support and promotion. Each individual manager or project leader must make a conscious decision as to the degree these guidelines will be imposed on an organization or project, or the appropriate "tailoring" of these guidelines to the organization, project, and environment at hand. The Sandia Software Quality Assurance Division may be consulted in making such decisions. Additional references may be required to provide the technical details and design principles that create the successful design and implementation environment.

1.4 Organization

Section 2 of the manual deals with software quality activities in general and describes why the structured approach of a Software Quality Assurance Plan (SQAP) is recommended. Most of the guidelines are presented in Section 3, which gives details on producing an SQAP at Sandia. The subsections of Section 3 are based on the SQAP subsections advocated by IEEE [IEE86]

and are grouped according to the quality plan framework criteria suggested in the Sandia *Quality Plan* [SQP86].

Section 4 describes how to encourage Sandians to follow a software quality plan, while Section 5 addresses the completeness of the SQAP. Finally, Section 6 describes how to change a software quality assurance plan.

Appendix A is an example plan for a hypothetical Sandia project. References for additional information are marked throughout the volume by brackets in this manner: [IEE86], with the details on these references listed in Appendix B. Appendix C provides a glossary of terms. For follow-on investigation, the book references are available through the Sandia library system, and AT&T videotapes on software quality are available from the Sandia Computing Education Center.

1.5 How to Use This Manual

This manual was designed to assist in creating a departmental, division, or project software quality plan. The user can turn directly to Appendix A and get an idea of the contents of an SQAP. However, this is but one example of an SQAP. A user's plan may need considerable modification depending on the scope, complexity, and criticality of the application involved. Suggestions on producing each portion of a plan can be obtained by reviewing Section 3, or by contacting Sandia's Software Quality Assurance Division.

This manual is designed to be used with the Sandia Software Guidelines, references [SSGv2] through [SSGv5], which provide details on suggested software quality practices at Sandia National Laboratories. In addition, IEEE references may prove helpful. The IEEE standard [IEE84] provides requirements for preparation and content of Software Quality Assurance Plans, while the IEEE Guide [IEE86] gives some detail on these plans. The IEEE references are available through Sandia's Design Information Center or from the IEEE Computer Society. The IEEE also provides a bound publication of all of their current software engineering standards.

2 Software Quality Relationships and Activities

Software quality assurance is achieved by using established guidelines for quality control to insure the integrity and prolonged life of software. Often there is some confusion as to the relationships between quality assurance, quality control, and the auditing function.

2.1 Quality Assurance, Quality Control, and Audits

A quality software product is one that

- satisfies the customer's requirements,
- meets the users' expectations,
- has the appropriate level of reliability, and
- has a logical design and understandable code for ease of maintenance.

Software quality assurance is a planned effort to provide confidence that a software product possesses the above criteria as well as additional attributes unique to the project such as portability, efficiency, reusability, and flexibility, while dealing with constraints based on time, manpower, or hardware. Quality assurance is the collection of the activities and functions that are used to monitor and control a software project so that specific objectives are achieved with the desired level of confidence. SQA is not the sole responsibility of Sandia's Software Quality Assurance Division but is determined by the consensus of the supervisor responsible for the project, the project leader, the project personnel, the users, and the SQA division.

The objectives of software quality are typically achieved by following a software quality assurance plan that states the methods the project will employ to assure the documents or workproducts produced and reviewed at each milestone are of high quality. Such an explicit approach assures that quality is not left to the "good intentions" of the individual project personnel. Although the integrity and capability of the software professional are the foundation of the project's quality, a plan manages the actions of the project personnel to confirm that all steps have been taken to achieve software quality, and provides management with documentation to confirm those actions. The plan states the criteria by which quality activities can be monitored rather than setting impossible goals such as "no software defects" or "100% totally reliable software."

Quality control is a part of quality assurance. Quality control consists of well defined checks on a product called out in the QA plan for the product. For software products, quality control typically includes reviews of specifications, inspections of code and documents, and checks for existence of user deliverables. Usually, document and workproduct inspections are conducted at each life cycle milestone to demonstrate that the items produced are within the criteria specified by the SQAP. These criteria are normally provided in the requirements specifications, preliminary and detailed design documents, and the test plans. The documents provided to the users are the requirement specifications, the preliminary and detailed design documentation, the results from the user acceptance test, the software code, the user's guide, and the operations and maintenance guide. Additional documents can be specified in the SQAP.

Quality control can be provided by various sources. For small projects, the project personnel's peer group or the department's Software Quality Coordinator can inspect the documents. On large projects, a configuration control board may be responsible for quality control. The board may include the users or a user representative, a member of the SQA division, and the project leader.

Audits are one of the traditional functions of quality control. Audits are independent examinations to assess compliance with some stated criteria. For software projects, audits may occur at two levels:

1. Audits of project compliance with the SQAP to determine all the quality control checks have been made.

2. Audits of some phase of the project to verify requirements established by the previous phase. For example, an audit of the code against the design documentation and standards, or an audit of the design against the requirements document. Such auditing activities constitute a *verification* review, as described in Section 2.2.1.

Audits are used to examine the software project for adherence to the written project rules at a project's milestone and at other times during the project's life cycle as deemed necessary by the project leader or the SQA personnel. An audit may be a detailed checklist for assessing compliance or may be a brief checklist to determine the existence of deliverables such as documentation. An audit report goes to the project supervisor, project leader, and project personnel to act upon. The report states the audit's purpose and the deficiencies found.

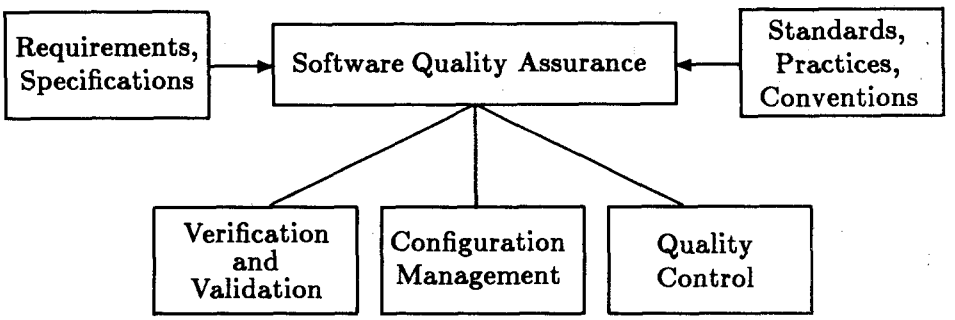

Figure 2: Elements Necessary for Software Quality Assurance

Who conducts the audits is explicitly stated in the SQAP. For small projects, the project leader or the department's Quality Coordinator can perform the audits. If the project is large, a member of the SQA division may lead an audit performed by an audit team. The team has a structure similar to the configuration control board discussed above under Quality Control. Following the audit, project personnel are assigned to correct the problems within a timeframe allotted for the correction. At Sandia, quality control functions other than audits are usually performed by project management personnel.

2.2 An Overview of Quality Assurance Activities

There is considerable debate about how a high degree of software quality may best be achieved. A growing body of literature and experience provides many common elements of successful software quality assurance programs. Functionally, most of these elements may be grouped into five categories, as shown in Figure 2.

As shown in Figure 2, most software quality assurance activities may be categorized as Verification and Validation, Configuration Management, or Quality Control. But the success of an SQA program depends also upon the existence of a well-thought-out, coherent collection of standards, practices, conventions, and requirements and specifications.

2.2.1 Verification and Validation Activities

The major purpose of verification and validation activites is to ensure that software design, code, and documentation meet all of the requirements imposed upon them. Requirements may exist as organizational or project-level conventions, standards, and practices; user requirements; specifications derived from and designed to meet user requirements; code review and inspection criteria; test requirements at the modular, subsystem, and integrated software levels; and acceptance testing requirements for code after it is fully integrated with hardware.

An example of an organizational convention could be "All code that is used by, or is used to provide information to, individuals outside this organization shall be written in 'C'. No module shall exceed 100 instructions in length and no module shall have a complexity greater than 8 as measured by Mc Cabes' Complexity Analyzer." Code reviews and complexity analyses can validate that project-generated code meets these requirements.

Lastly, the extent to which personnel independent of the design and coding should participate in SQA activities is a matter of institutional, organizational, and project policy. SQA Division approval of verification and validation plans is required for some code. Examples would include software used in the development, manufacture, operation, or test of weapon components or WR systems.

In brief, given that there are good reasons for the requirements on the software and on its associated documentation, verification and validation are activities to assure that these requirements are met. During software design and implementation, verification is needed to determine whether the products of one phase of the software development life cycle fulfill the requirements established during the previous phase. The verification effort is less time-consuming and is less complex when conducted throughout the development process. Validation is a software evaluation at the end of the software development cycle.

2.2.2 Configuration Management Activities

The second principle element of SQA shown in Figure 2 is Configuration Management. This term refers to those activities designed to assure that design and code are well-defined and cannot be changed without a full review of the impact of the change and full documentation of the change. The basic purpose of configuration management is to control code and its associated documentation so that final code and its description are consistent and

represent those items that were actually reviewed and tested. Thus spurious changes to "make the code better" are eliminated.

As an example, after a software module is thoroughly reviewed and tested, it is considered *baselined* and is given an "issue" or version number. A separate copy of that issue is controlled by management. The need for any proposed changes to that issue should be carefully reviewed for need versus impact (necessity of the change versus cost of retesting). This process is discussed in some detail in Volume 4, *Configuration Management* [SSGv4].

The same control concept is applied to design to assure that it is "frozen" after being reviewed and approved. When requirements change, configuration management activities may be used to control changes to design and delete any "obsolete" code. There are clear advantages to configuration management activities in large, multi-programmer projects, and in long-term single-person projects.

As with verification and validation activities, the general nature of configuration management efforts may be a matter of organizational policy, while specific implementation of this policy may be described at the project level. For example, an organization may require that a separate configuation management plan be prepared for projects that will generate more than 20,000 lines of code, and that a configuration control board (with the types of members specified) should exist for a large project. At the project level, the response would be to define such a configuration management plan and control board, structured so that both meet organizational requirements.

2.2.3 Quality Control Activities

The basic, "traditional" function of quality control is to verify that the other quality assurance functions are performed properly and in a timely manner as required by organizational and project requirements documents. In this area, the issues dealt with are illustrated by the following checklist:

- ☐ Was the review conducted?
- ☐ Did the required people attend?
- ☐ Were all of the planned review activities completed?
- ☐ Was a report written?
- ☐ Were all post-baseline changes reviewed by the Configuration Control Board, controlled in accordance with configuration management requirements, and properly documented?

- ☐ Were tests performed at the modular (or any other) level consistent with the requirements?

- ☐ Were any requirements not tested?

These traditional quality control functions are typically identified as **audits**. The purpose of audits is to assess compliance with requirements, standards, and practices, to confirm the existence of all sections of a project's documents, and to measure the project's progress. The term "audit" also refers to activities undertaken to assess the degree of adherence of an organization to its own, and institutional, quality requirements. Such audits are typically conducted by a designee of organizational management (*e.g.*, the quality coordinator for the organization) and by the Quality Operations Division, independently.

In a highly controlled project, the actual *control* functions of quality control might include such things as the capability to prevent further activity in the project life cycle until all (or some critical subset of) requirements established during the previous software life cycle phase are met, or to prevent release and use of the final product until all requirements on the code and its documentation are met. At Sandia, these control functions are often reserved for the project leader or supervisor, so that these persons perform quality control functions, while personnel performing the audit functions illustrated above serve to identify areas in which control might need to be exercised.

In addition to the audit role and the possible exercise of direct control over development activities, quality control personnel are sometimes given the responsibility for some of the verification and validation and configuration management functions. For example, if individuals with quality control responsibility are software engineers, they may be given full responsibility for developing the software verification and validation plan (sometimes in concert with the Software Quality Assurance Division, depending upon requirements), conducting some or all of the software tests, and leading software review efforts. They may also be assigned configuration control board responsibilities.

2.3 Requirements to Implement an SQA Program

The first clear requirement to implement software quality assurance programs is that organizations and projects must define the conventions, practices, and standards that are applicable to their activities. The most essen-

tial of these encompass software design, software implementation, software verification and validation, configuration management, and quality control.

Internal, organizational standards might address which reports, documents, manuals, etc., are to be prepared; what format to use; who is responsible (generically); what level of quality control is required as a function of project complexity; and what overall standards for design, implementation, testing, etc. are to be applied to projects for which the organization is responsible. An individual project, on the other hand, would expand upon such requirements, respond to such requirements, or add its own.

Considerable care should be exercised in the definition of standards, activities, and responsibilities, both at the organizational level and the project level. Ill-conceived requirements will not lead to the desired high-quality software. The requirements should stem from the quality objectives based on the activity and the personnel associated with it. Generally, it is the responsibility of the organization or project to decide how to achieve its own desired software goals, using guidance such as this document. For critical software that may affect the quality of a weapon, weapon component, or associated data or information, the Software Quality Assurance Division should be consulted for additional guidance.

A second requirement is that organizational and project policy and approaches be documented. Sandia's *Quality Plan* [SQP86] suggests the same need. One effective approach is to write a software quality assurance plan which specifies participants, their responsibilities, the nature of software quality assurance activities, as well as the character of both external and project-developed specifications and requirements. The major purpose of a software quality assurance plan is to provide definition of the activities necessary throughout the software life cycle. Section 3 discusses the planning process in more detail, and Appendix A provides an example project plan.

2.4 Why have an SQA Program?

Software quality assurance programs may at first appear to be burdensome and unneccessary. "Why not just write the code?" is a common question. Many of us just write the code on a personal computer to support our own activities of scheduling and planning. It is not unreasonable to ask why this is inappropriate in other areas.

All too often the planning and design steps that keep a project under control are thought to be expendable. What looks like a short-cut becomes expensive in quality, maintainability, and even time.

Preparing and using an SQAP allows control throughout the life of the software. Two important project aspects that can help demonstrate this point are software design and project maintenance.

Software design and implementation can be enhanced by the stated standards and guidelines in the SQAP. A typical standard is to have a particular task coded in only one place. This greatly simplifies tracing through the control paths. Other standards are the use of well-defined program states, and accurate in-line comments. The software can be easily evaluated to determine whether it fulfills the requirements.

Code maintenance also has associated problems. Maintenance allows for the continued use of software. Software development without standards and documentation makes determination of what a code segment is doing difficult and time consuming, if not impossible. Even if modifications can be made, unwanted side-effects may go undetected. Many times, software has been re-written from scratch because it was not possible to modify it. Control over code maintenance should be established by the same configuration management activities that were imposed during code development.

A software quality assurance plan provides the framework and guidelines for the development of understandable and maintainable code. These ingredients help to ensure the correctness and quality sought in a software project.

3 Contents of a Software Quality Assurance Plan

A software quality assurance plan provides the procedures for assuring high quality software produced or maintained in-house or under contract. These procedures affect the areas of planning, designing, writing, testing, documenting, storing, and maintaining computer software. The following describes the contents of an SQAP, and relates these contents to the specific sections prescribed by the IEEE Standard for Software Quality Assurance Plans [IEE84]. Appendix A exemplifies a plan with sections written according to the standard.

3.1 Introduction and Program Structure

An SQAP should give the formats, administrative aspects, rules, and requirements to follow, and where possible, an outline of content for all required documents. The SQAP should:

- list (*not* describe) tools and methodologies
- outline project responsibilities (*not* describe how to design)
- reference (*not* detail) software development guidelines

because an SQAP assures the quality of the software rather than specifying procedures for developing and maintaining the software.

3.1.1 Purpose

The purpose of an SQAP (provided in Section 1 of an SQAP according to [IEE84]) is to provide a set of requirements that, when met, assure that the software satisfies its intended mission, is useable, and is maintainable. An SQAP can be applied to any software developed at Sandia National Laboratories.

At Sandia National Laboratories, SQAP's are implemented on a project-by-project basis as decided by the project supervisor (*i.e.*, by management at the supervisor level or higher). The project leader (usually a staff member) will use these guidelines and may work with the Software Quality Assurance Division to write and implement the SQAP.

According to the IEEE Guide for Software Quality Assurance Planning [IEE86], the SQAP should address the following questions:

1. *Which software products are covered by this SQAP?* Name each product.

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

3. *Why is this SQAP being written?* Is the plan being written in response to internal or external requirements? Why is the plan needed?

4. *What documents form the basis of this SQAP? If there is deviation from these documents, what is the rationale?* Describe the extent to which the SQAP is based on this volume or on the IEEE Standard for Software Quality Assurance Plans [IEE84].

3.1.2 Reference Documents

The project leader provides (in Section 2) a list of documents that may be helpful in following and implementing the SQAP. The sample plan in Appendix A gives examples of the types of documents that may be needed. Appendix B provides bibliographic information on some actual reference documents.

3.1.3 Management

The project supervisor is responsible for deciding whether an SQAP will be applied to a particular project. If an SQAP will be applied, the project leader will direct its development and implementation in cooperation with the Software Quality Assurance Division. This plan (in Section 3) specifies the project software quality assurance tasks and responsibilities.

3.1.3.1 Organization

The IEEE standard [IEE84] states that an organizational paragraph in the SQAP should depict the organizational structure that influences the quality of the software and should include a description of each major organizational element involved with the software project. Organizational dependence or independence of the elements responsible for SQA from those responsible for software development should be clearly described or depicted. [See Appendix A for an example of this].

3.1.3.2 Responsibilities

Finally, the SQAP will detail the responsibilities of each organization involved in the project. Both software development responsibilities and overall management responsibilities should be included. Because different software projects vary so widely in size, the separation of the SQA responsibility from software development is at the discretion of the project leader.

3.2 Design Control

This section identifies quality program elements for the three areas which define Design Control according to Sandia's Quality Plan [SQP86]:

- Design Information – Maintain engineering documentation and standards, practices, and conventions to assure that the software is developed to the latest applicable requirements.
- Design Review – Review the products to verify that they are adequate to fulfill the specific intent and have been appropriately validated.
- Change Control – Manage configuration, report problems, and identify corrective action to assure that changes are governed by consistent control measures which allow changes to be implemented in a timely manner.

3.2.1 Design Information

3.2.1.1 *Engineering documentation* is a documentation set that describes activities in the software life cycle from development through use and maintenance of the software product. Section 4 of the SQAP identifies the documents required and describes how to assess document adequacy. The Guidelines for *Documentation* and *Standards, Practices, and Conventions* ([SSGv2], [SSGv3]) contain detailed information on each of the following document types, including its purpose, content outline, and intended audience.

- Project Plan
- Software Requirements
- Software Design Description
- Results of Reviews and Audits

- Software Verification and Validation Plan and Resulting Report
- Test Set Documentation and Results Report
- Maintenance Documentation
- Training Plan
- User's Manual and Operating Procedures

Most of the documents will not be static, such as the project plan, maintenance, and test results documentation. Design and requirements documents should be up-to-date as well.

3.2.1.2 The SQAP describes (or references) in Section 5 *standards, practices, and conventions* to be followed for a project. Examples include:

- variable and file naming conventions
- version designations during development and test
- software prologue content

3.2.1.3 The SQAP identifies (or references) in Section 9 special *tools, techniques, and methodologies* used for a project. Different tools and techniques are suitable for different phases of the software life cycle. Refer to Volume 3, *Standards, Practices, and Conventions* [SSGv3] for suggestions of tools by phase, and Volume 5, *Tools, Techniques, and Methodologies* [SSGv5] for a directory of software tools and methodologies at Sandia.

3.2.2 Design Review

Section 6 of the SQAP identifies the reviews and audits to be performed, and provides guidelines for conducting them. Reviews and audits are used to determine compliance with specifications and standards, and to assess progress made in the project. Deviations from specifications and standards caught early in the software life cycle are easier and thus cheaper to correct. Another benefit is increased communication between project members, management, and others involved in the reviews and audits. The difference between audits and reviews is the following: An audit is an independent examination to assess compliance with software requirements, specifications, and standards. A review attempts to uncover design or code deficiencies affecting system quality. A review may be performed by the QA group,

the project leader, developers, users, or code testers. Administration of the following customary reviews is discussed in Volume 3, *Standards, Practices, and Conventions* [SSGv3].

- software requirements review
- software design reviews - preliminary and detailed (or critical)

3.2.3 Change Control

Configuration management helps protect the investment made in software development, maintenance, and project documentation. Sections 3.3 and 3.4 contain further details. This SQAP section (Section 7) is to:

- Identify methods of protecting versions of completed code, as well as documentation.
- Identify who is responsible for various facets of configuration management.
- Identify who is authorized to make changes.
- Identify who is authorized to approve changes and who comprises the configuration control board, if there is one – supervisor, project leader, etc.
- Describe the interface control, *i.e.*, the management of changes which affect several projects.
- Identify the product baseline.

Refer to Volume 4, *Configuration Management* [SSGv4] and the configuration management section in Volume 3, *Standards, Practices, and Conventions* [SSGv3] for further details.

Following the configuration management section, section 8 of the SQAP should *identify methods for reporting, tracking, and fixing software problems* during development and maintenance. The purpose of problem reporting and resolution is to:

- Document problems
- Determine validity of reported problems

- Provide a vehicle for developer, user, and manager to assess problem status
- Provide feedback for assessing software quality and reliability

3.3 Other Quality Factors

This area of the SQAP covers the more administrative aspects of the plan. It addresses the questions of

- Code Control. Where is the code located? Who has access to it, and what procedures do they need to follow for reading or modifying the code?
- Media Control. What medium is the code on? How is it backed up? How is it stored?
- Supplier Control. What should you ask of a supplier in terms of quality assurance? What can you expect to get?

3.3.1 Code Control

Section 10 of the plan describes how the actual programs will be managed. The Configuration Management section of the plan will indicate things such as what constitutes a baseline, who authorizes and approves changes, and how portions of code relate to one another (interface control). The Code Control section, on the other hand, defines how the policies identified in the Configuration Management section are carried out. This section should do the following:

- identify the criteria which will determine how, and to what degree, a particular module will be controlled. For example, utility routines called by many modules may need broader review of anticipated changes than single-purpose modules. These criteria can then be used by the design team to specify controls on a module-by-module basis.
- prescribe the conventions for labeling and cataloging the controlled software. Should directory listings be glued to a tape reel? To a disk-pack cover? A commercial database manager may be chosen to keep records of, *e.g.*, which releases of which modules are in what cabinet, on which optical disk. It should be decided to what degree of detail it is necessary to specify these sorts of conventions. The conventions

then should be documented in the plan. (The Department of Energy has regulations for the cataloging and labeling of media according to classification – these should be included as part of the SQAP if appropriate.) Who will do all of this labeling and cataloging? If a librarian is required, that requirement should be stated in the plan.

- identify physical storage location requirements for software based on criteria such as need for immediate access, need for protection of critical or classified code, etc. This is not to say that a good terminal I/O routine must be hidden from those who may wish to borrow the code, but neither should a tape reel containing the only copy of the world's fastest sort program be left around where it might be used as a scratch tape for backups (again, include DOE and security requirements if and as appropriate).

- list procedures for distributing copies, where such copies are authorized by the Configuration Management Section. The impact of having "unreleased" code in circulation should be assessed. The degree of stringency in code protection should be specified in the plan.

- identify the documentation affected by various levels of change. For example, if a bug is fixed to make the code work as originally documented, there may be no documentation change required (except, of course, for those documents created as a result of the change-control process). If a simple instruction sequence is replaced with an obscure one to tune the system for better performance, the maintenance documentation should reflect the change, but the user's guide should need no modification. If enough features are added to make the user interface significantly different, the user's guide, reference manual, code documentation, etc., should be changed.

- describe procedures for implementing a new version. Such procedures might cover all of the labeling, cataloging, and documenting described above, as well as distribution lists and procedures.

3.3.2 Media Control

Section 11 deals with physical protection and control of the media, such as tapes, disks, punched cards, firmware, etc., containing the code and documentation, to ensure that the Code Control and Configuration Management can be implemented effectively. To some degree, the issues of 3.4.1 apply

here, since physical protection of the media is intimately associated with protection of the code contained therein. The SQAP may need to address the following issues:

- means of physical access for software storage and retrieval. If the development machine on which work is to be done is password protected, the plan should address password control. If the machine, terminals, and all storage media are in a locked room, the plan may address access control for the room. Are backups a routine part of the operation of the machine, or will they need to be done specifically for this project? This question should be resolved in the plan.

- physical storage precautions to be taken to protect code with special access or protection requirements as defined by Configuration Management or Code Control. The DOE, again, gives detailed guidance in the protection of classified materials. The plan also may include special physical protection, such as the locked room for code which the specifiers feel needs to be restricted to the developing and maintaining organizations.

- environmental precautions to be taken, where appropriate, to ensure integrity of the data on the storage media.

3.3.3 Supplier Control

Section 12 describes the steps to be taken to assure quality of software purchased from outside sources. The avenues available for SQA will vary depending on how intimately the supplier wishes to interact with Sandia or the DOE. Note that there will probably be a significant difference between dealing with contractors developing software specifically for Sandia, and vendors supplying software "off-the-shelf".

- If possible, the plan should include requirements for the supplier to provide evidence of adherence to an SQAP, especially for contractor-developed software, such that

  - Sandia has an opportunity to augment the supplier's plan to include items necessary to the project. Such extension will clearly never be the case for a commercially available, "off-the-shelf" product.

  - means are provided to monitor administration of the SQAP. Such a requirement is also not applicable to an "off-the-shelf" product.
  - remedies are specified for non-compliance to the supplier's plan.

- Realistically (especially for vendor-supplied software) there will often be severe limitations on the ability or willingness of vendors to comply with Sandia's SQA requirements. In these cases software should be considered much like other purchased commodities, and the project SQAP should include considerations such as

  - how well do the specifications of the software meet the needs of the project?
  - how well does the software adhere to its specifications? Often, a supplier eager to tap a potentially lucrative market like Sandia will loan out software (and even hardware) for evaluation. In these cases, a test and evaluation plan for purchased software may need to be included in Sandia's SQAP.
  - what kind of warranty does the supplier offer? Is the supplier willing to remedy non-compliance with the specifications promptly, or at all?
  - what kind of support (installation, help, upgrades) does the supplier offer?
  - is the source code available? Is the documentation even available?
  - what is the reputation of the vendor? Of the product?

Attention to these administrative matters at the outset will make the project less hectic and ultimately more rewarding for everyone involved.

3.4 Documentation Control

Section 13 identifies project records to be maintained and specifies the retention periods. The records include:

- Technical reports, *e.g.:*

  - user's manuals
  - maintenance manuals

- Configuration control records for the software

- Inspection and test procedures
- Results of evaluations, inspections, reviews, audits, and tests of development and production software
- Formal project documentation, *e.g.:*
  - system specification
  - preliminary design specification
  - detailed design specification
- Project log books or notebooks

All project documentation must be kept *current*, including requirements, design, and test documentation. Thus, control features for assessing timely issuance and updating of such records should be described. The format desired for the project records should be specified, and the intended audience for each identified.

3.4.1 Records Collection

The *type* of records to be collected on a Sandia project are determined by the *objectives* for record collection. The records to be collected should be based on the quality assurance requirements with the overall objectives documented in the SQAP. Examples of records collected for certain categories of projects are given in Table 1 below.

An objective is to provide evidence that the software development process was performed in conformance with established professional practice, or the customer's requirements, particularly for a reimbursable project. Another objective of records collection for any project is to keep management informed of development progress. A third objective of records collection is to provide historical information for future repetitive testing during project software changes. This historical information should include reports of corrective action – what action was taken, what components were changed. Development and test records must be collected, as well as documentation on tools used to develop and test the software and the system.

3.4.2 Records Maintenance

Records are maintained to furnish evidence of the quality of the computer programs and associated documentation, and to record how the programs

Type of Project:	*Example Records:*
Weapon System (*i.e.*, WR) Project	Requirements document "SD" (Software Documentation) Design Document Test Plan User's Manual
Reimbursable Project	Requirements document Design Document Results of reviews and audits User's Manual
In-house Project	Requirements document Design Document Maintenance Manual

Table 1: Example Records for Collection

were developed so that evaluation by the users and management can be performed. The records include the results of reviews, inspections, tests, and audits. Closely related data must also be maintained, such as data related to project procedures and equipment. The project leader ensures that all these data are systematically collected and analyzed and the results used for the detection, correction, and prevention of deficiencies. This collection of data includes error reporting data, such as software nonconformance reports or problem reports.

The plan specifies the record maintenance and review schedule, and identifies the record media, such as electronic files, hardcopy, or microfiche format. The individual responsible for maintaining the records should be named by position, *e.g.*, the project librarian or project manager. Procedures to enforce parallel updating of documentation and software code changes are referenced here. Example: "The project librarian only accepts code into the library which is accompanied by the corresponding documentation."

3.4.3 Records Retention

The plan specifies the length of time the project needs to retain each type of record, once the project has been completed and the system has been

delivered. At Sandia, the guidelines in Table 2 (below) are suggested for the project records itemized in this section.

Type of Project Record:	*Retention Period:*
Technical (*e.g.*, SAND) Reports	indefinitely
Configuration control records	life of system
Inspection and review results	six months to life of system
Test results (one complete set, plus latest change results)	life of system
Formal project documentation	life of system
All other project documentation	one year to life of system

Table 2: Records Retention Periods

Additional details and suggestions for documentation control are found in Volume 4, *Configuration Management* [SSGv4].

4 Implementation of a Software Quality Assurance Plan

The implementation of an SQAP involves the process of planning, formulating, and drafting the contents of an SQAP. Implementation also involves obtaining the necessary approvals for the plan as well as developing a plan for executing the SQAP. The subsequent evaluation of the SQAP (described in Section 5) will be performed as a result of its execution. This implementation process can be divided into four categories:

1. Acceptance by Management
2. Acceptance by Development Personnel
3. Planning for Implementation of the SQAP
4. Execution of the SQAP

4.1 Acceptance by Management

Management participation is necessary for the successful implementation of an SQAP. Management is responsible for both ensuring the quality of a software project and for providing the resources needed for software development.

The level of management commitment required for implementing an SQAP depends upon the scope of the project. If a project spans organizational boundaries, approval should be obtained from all chains of affected management (*e.g.*, Department 2640, Division 9812). Once approval has been obtained, the SQAP is placed under configuration control (see Section 6). In the management approval process, management relinquishes tight control over software quality to the SQAP administrator in exchange for improved software quality. By and large, Sandia's software quality has been an issue left to software implementors. High quality is desirable, but concern may be expressed at the management level as to the "cost" of a formal software quality assurance program. Staff should be aware that management views an SQAP as a means to ensuring software quality, not as an end in itself.

In order to address management concerns, software life-cycle costs should be formally estimated for projects implemented both with and without a formal SQAP. In general, implementing a formal SQAP makes good economic and management sense.

4.2 Acceptance by Development Personnel

Since the software development and maintenance personnel are the primary users of an SQAP, their approval and cooperation in implementing the plan is essential. The software project team players will be required to adhere to the project SQAP, so all team players must accept the SQAP and follow it.

No SQAP will be successfully implemented without the involvement of the software team members and their management in the development of the plan. At Sandia, project teams generally consist of only a few members. In this case all team members should actively participate in writing the governing SQAP. When projects become much larger (*i.e.*, encompassing entire divisions or departments), representatives of project subgroups should provide input to the development of the SQAP. Constant feedback from representatives to team members will help gain acceptance of the SQAP.

4.3 Planning for Implementation of the SQAP

The process of planning, formulating, and drafting an SQAP requires staff and word processing resources. The individual responsible for implementing an SQAP must have access to the appropriate resources. In addition, the commitment of resources requires management approval and, consequently, management support (see Section 4.1). In order to facilitate resource allocation, management should be made aware of any project risks which may impede the implementation process (*e.g.*, limited availability of staff or equipment). A schedule for drafting, reviewing, and approving the SQAP should be developed.

4.4 Execution of the SQAP

The actual process of executing an SQAP by the software development and maintenance team involves determining necessary audit points for monitoring the SQAP. The actual evaluation of an SQAP is the subject of Section 5. The auditing function must be scheduled during the implementation phase of the software product so the software quality assurance program will not fail due to improper monitoring of the software project. Audit points should occur at either periodic points during development or at specific project milestones, such as major reviews or a project deliverable.

5 Evaluation of a Software Quality Assurance Plan: Guidelines

A draft Sandia engineering procedure on Software Quality Assurance for Sandia Designed or Developed Software ([EPd87], based largely on [IEE86], the IEEE Guide for Software Quality Assurance Planning), calls for the preparation of a software quality assurance plan for the design and development of software that requires certification. The procedure will define Sandia policy that will assure that software designed or developed has well-defined objectives and has been adequately documented, reviewed, and tested to assure that the objectives and requirements are met.

Software quality assurance should make provisions for periodic or ongoing evaluation of the SQAP. Such evaluation can be conducted by the project leader or, for critical software, by a member of the SQA division. Evaluation of the SQAP involves examining it from two viewpoints:

- the plan's contents
  - has a clear statement of the scope and applicability (complexity) of the plan
- the use and administration of the plan
  - determines what is in the plan and how the plan is administered

5.1 Evaluation of the Plan's Contents

The evaluation should be done based on a checklist which allows the evaluator to verify that all requirements have been addressed.

Thus, the SQAP should have a purpose stated, a description of the software, and what checkpoints are to be used. The assessment of the SQAP can be accomplished by asking a series of questions:

- What are the mandatory software requirements (*e.g.*, IEEE, Sandia, Nuclear Regulatory Commission) and are they addressed?
- What other standards are referenced?
- Are exceptions adequately justified?
- Is the content of the SQAP sufficient to achieve the stated objectives?

The evaluator should create or specify a form for evaluation that will address these basic questions, as exemplified in Table 3.

Required SQAP content:	*Where found in SQAP:*	*Evidence that plan is adequate:*	*Exceptions noted:*
Specific purpose and scope described			
Software product item described			
Tools, techniques, methods: documented, justified			
Standards, practices, and conventions stated			
Requirements and specifications stated			
Computer Program Design: documented, complete			
Documentation: procedures, delivery to end-user			
Computer Program Library: procedures, controls documented			
Reviews and Audits: preparation, execution			
Configuration Management: how CM objectives are assured			
Testing: states reviews of requirements, criteria, plans			
Corrective Action: provides prompt correction of defects			

Table 3: Example SQAP Content Evaluation Form

5.2 Evaluation of the Plan's Administration

The SQAP administration is to be evaluated in terms of the availability of relevant documents, procedures, and plans. For example, one might evalute the plan as to how adequate the procedures are for establishing traceability of requirements and for verifying that management monitoring and control is adequate and timely.

Here also the evaluator should create or specify an evaluation form that will address the administration of the plan, as exemplified in Table 4.

Requirements	*Responsible Person*	*Verification Method*[1]	*Verification Frequency*
Management activities completely described			
All referenced documents available			
Software documentation complete			
Software Configuration Plan exists			
Procedure for problem reporting and corrective action in place			
Code control procedures available			
Tools, Techniques, and Methodologies exist			
Records Management System in place			

Table 4: Example SQAP Administration Evaluation Form

Each organization, at whatever level is appropriate, should identify a responsible person for the SQAP evaluation function.

At Sandia, designated personnel writing an SQAP for a project have another resource available to them. The Software Quality Assurance Division may provide guidance in formulating plans and useful feedback in reviewing plans. The Quality Operations Division can also perform independent internal audits of the SQAP.

Note[1]*: Verification should be in sufficient detail to assure traceability. Examples: citing documents and procedures which are in place, Memo For Record of activity completed, etc.*

6 Modification of the Software Quality Assurance Plan

Even the best SQAP is no good if it is hopelessly out-of-date with the software product. An approved and implemented SQAP can and should be modified to reflect the quality objectives of the current software product.

The questions of when and why a plan should change are answered in subsections:

- Keeping the Plan Current
- Reasons for Changing the SQAP

The questions of who should change the plan and how to change the plan are answered in subsections:

- Formality of Change Control
- Methodology for Changes

6.1 Keeping the Plan Current

The plan should be updated whenever the project has an approved change in requirements or environment. At each milestone in the project, the plan should be reviewed for evaluating its relevancy to the current project and preparing for the next stage of the project's life cycle. The plan is a viable working document since it describes the quality control imposed upon the software product. Since it is an approved document, the SQAP should be controlled. The degree of control varies, dependent on the project. Formal configuration management control may be necessary for a large or critical project while informal control such as a page in the project's workbook or file folder may suffice for less critical or smaller projects. Clearly, whenever changes are approved, all the project team members should receive a copy of the revised plan.

6.2 Reasons for Changing the SQAP

A project's SQAP can be changed for many reasons, using the previous section (Section 5, Evaluation of an SQAP) as a guide. The following examples are possible reasons for changing a plan:

- The project's environment has evolved because of
  - changing project objectives or deliverables
  - hardware changes;
- The project's management has altered because of
  - changing responsibilities of the project or SQA personnel
  - changing participants;
- The plan has omissions, and needs
  - more or better SQA techniques (tools)
  - amplified test sets;
- The plan has identifiable deficiencies
- The plan has extraneous information, such as
  - sections that no longer apply
  - outdated standards, practices, or conventions.

6.3 Formality of Change Control

If the project is large, lengthy, or has "critical" factors, then it may benefit from having a more formal organization, *e.g.*, a **configuration control board**, to rule on changes to the plan or the project. Such a board can be created on an ad hoc basis as needed or appointed for the duration of the project. The membership of the board could include (but is not limited to):

- project leader
- systems analyst
- hardware engineer
- software team leader
- SQA specialist
- customer representative
- user representative

- independent verification and validation personnel
- management representatives (for projects that span organizations)

For small projects, the project leader will be the "board." In addition to ruling on changes, a configuration control board can act as the quality controller (take action to approve the documents required) to perform the audits and to interface with the customers and users. The independent verification and validation (IV&V) source may work well if the personnel are not associated with the project, have worked on a similar project, and are familiar with SQA techniques.

6.4 Methodology for Changes

The configuration control board addresses problems in the plan and considers suggestions for improvement. Change requests can come from anyone associated with the project or possibly external people, *e.g.*, auditors or upper level management. The requests should be written, using a change control form similar to the error reporting form for software [SSGv3].

The board must decide if the requests are legitimate and pertinent to the SQAP. If they are, a modification of the document is in order. The steps are:

1. Seek several alternative solutions that may solve the problem.
2. Review all proposed solutions and determine what solution best answers the problem.
3. Get management approval of the revised plan.
4. Incorporate the changes in the SQA plan, highlighting the changes made.
5. Distribute the revised plan to all interested parties.
6. File and control the revised plan.

Note: For large projects or a potentially troublesome solution, the following actions are recommended between steps 2 and 3:

- Send the recommended solution to project and SQA personnel for their review and comments.

- Reevaluate the solution, reviewing comments received, and either approve, reject, or modify the solution, iterating on these last two actions as necessary.

Appendix A

Example Software Quality Assurance Plan

This Appendix provides a sample Software Quality Assurance Plan for a project at Sandia. For the purpose of this example, the details of the project have been obscured to emphasize the general philosophy and the techniques employed in the plan.

SOFTWARE QUALITY ASSURANCE PLAN

for

SANDIA NATIONAL LABORATORIES GENERIC PROJECT (GENERIC)

Version 1.0
August 13, 1987

Submitted By: Larry J. Marsupe
GENERIC Project Member
Sandia National Laboratories

Reviewed By: Sandra A. Raison
GENERIC Project Member
Sandia National Laboratories

Approved By: _____
GENERIC Project Leader

TABLE OF CONTENTS

SECTION		PAGE
1.0	Purpose	b
2.0	Reference Documents	c
3.0	Management	d
4.0	Documentation	f
5.0	Standards, Practices, and Conventions	i
6.0	Reviews and Inspections	i
7.0	Software Configuration Management	j
8.0	Problem Reporting and Corrective Action	j
9.0	Tools, Techniques, and Methodologies	j
10.0	Code Control	j
11.0	Media Control	j
12.0	Supplier Control	j
13.0	Records Collection, Maintenance, and Retention	k
14.0	Testing Methodology	k
APPENDIX	Glossary and Acronyms	k

GENERIC Software Quality Assurance Plan

1.0 Purpose

This Software Quality Assurance (SQA) document provides a plan for producing, controlling, and maintaining software for the Sandia National Laboratories Generic (GENERIC) project. The GENERIC project will provide a capability for the exchange of product definition data between different brands of interactive graphics systems. This SQA plan is written in response to and is in conformance with the DOE-CIM Standard CIM-101, *Management Plan for Quality Assurance of Software Shared Within the DOE.* This plan is based on ANSI/IEEE STD 730-1984 and the *Sandia Software Guidelines*, Volume 1.

The software products covered by this plan are the software programs PARSER, KERNEL, and FILEWRITER. This plan also provides guidelines for developing, documenting, and maintaining various TRANSLATOR software programs.

The intended use of this software is the preservation of visual equivalence of drawings between sending and receiving systems. The approach to the exchange process is based on the use of a standardized data format used by the sending system to put the data into a non-proprietary form where it can be manipulated as required through vendor specific translator software, and also by the receiving system at the completion of the exchange process.

Figure 1 presents a block diagram of the proposed Phase 1.5 GENERIC software system. It shows the software programs and the data transformations between software programs that are required for data exchange between two graphics systems.

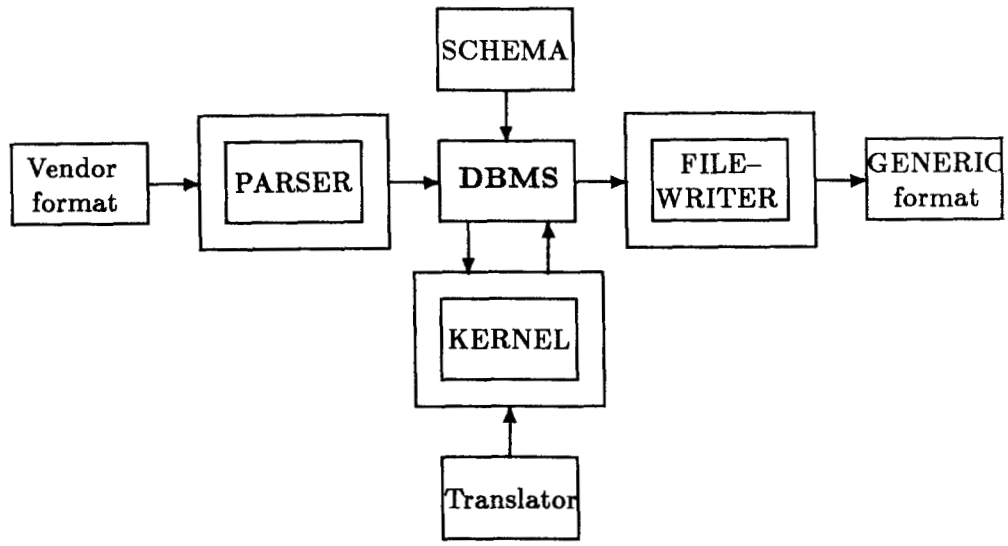

Figure 1: GENERIC System

The GENERIC project will provide a processing environment and auxilary software for automating file manipulations deemed necessary for successful transfer of product definition data. For the early phases (Phase 1.0, prototype; Phase 1.5, production), all such file manipulations are performed while the data resides in a relational database maintained through a database management system (DBMS).

2.0 Reference Documents

2.1 DOE-CIM Standard CIM-101

Management Plan For Quality Assurance Of Software Shared Within The DOE, DOE/AL, *draft dated* Sept 1986.

2.2 Sandia Software Guidelines

Volume 3, *Standards, Practices, and Conventions*, SAND85-2346, July 1986.

2.3 Software Requirements Specification

Software Requirements for The Sandia National Laboratories Generic Project (GENERIC), Phase 1.5, Sandia Org 9810, August 1986.

2.4 GENERIC Project Plan

The Sandia National Laboratories Generic Project (GENERIC) Administrative Plan, Phase 1.5, Sandia Org 9810, March 1987.

2.5 GENERIC Software Test Plan

The Sandia National Laboratories Generic Project (GENERIC) Software Test Plan, Phase 1.5, Sandia Org 9810, February 1987.

2.6 Software Configuration Management Plan

SCMP for the Sandia National Laboratories Generic (GENERIC) Project, Sandia Org 9812, *draft dated* Jan 1987.

3.0 Management

3.1 Organizational Structure

The GENERIC software project operates within the organizational structure diagrammed in Figure 2:

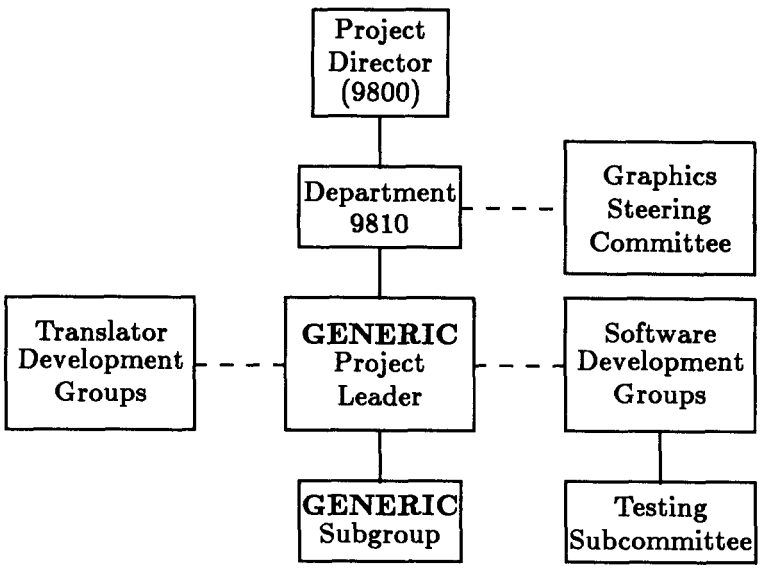

Figure 2: GENERIC Project Organizational Structure

3.2 Tasks and Responsibilities

This section identifies the responsibilities of specific individuals and organizations.

3.2.1 Project Leader (Lead Software Engineer)

will:

- Prepare and implement this Software Quality Assurance Plan.
- Prepare the Software Requirements Specification [Ref 2.3].
- Manage the GENERIC software development.
- Perform integration of the GENERIC software programs.
- Distribute the GENERIC software to requesting users and coordinate the software installation.
- Develop and implement configuration control procedures for the GENERIC software system.
- Retain the GENERIC software system acceptance test results.
- Coordinate the GENERIC system maintenance with users and Software Development Groups.
- Provide training in the development of TRANSLATOR software units.

3.2.2 Software Development Groups

The Software Development Groups (SDG) are defined as follows:

- Sandia National Laboratories, Division 9813, is responsible for the KERNEL software unit.
- Sandia National Laboratories, Division 9811, is responsible for the SCHEMA and PARSER software units.
- Commercial Contractor, Albuquerque, NM, is responsible for the FILEWRITER software unit.

For each software unit for which an SDG has responsibility, the SDG will:

- Develop documentation as defined by section 4.2 of this SQAP.
- Perform and document software unit verification testing.
- Provide the source code and test results summary to the Project Leader.
- Revise software units as directed by the Project Leader.

3.2.3 Testing Subcommittee

The Testing Subcommittee is comprised of one staff member from each of the SDG's and is chaired by the Project Leader. Its responsibilities are to:

- Prepare, coordinate, and implement the GENERIC system test plan.
- Develop and carry out integration and acceptance testing for the GENERIC system.
- Prepare system integration and acceptance testing results summary for the GENERIC system.

4.0 Documentation

This section identifies the governing GENERIC project documentation and states how these documents are to be checked for adequacy. Responsibility for a document implies origination, verification, maintenance, and control of the document.

4.1 Software Requirements Specification

The purpose of the Software Requirements Specification [Ref 2.3] is to serve as the instrument of record for the requirements and approvals for the GENERIC Software System. The Project Leader is responsible for the SRS; the document will follow the specified outline for an SRS as provided in *Sandia Software Guidelines*, Volume 2," Documentation." The qualities and contents of this document will follow ANSI/IEEE Std 830-1984, *IEEE Guide to Software Requirements Specifications*, and will be inspected during the Software Requirements Review as identified in Section 6.

4.2 System User's Guide

The User's Guide will provide:

1. Background information.
2. Instructions for use of the system.
3. System limitations.
4. System error handling.
5. Input/Output requirements.

6. Input/Output examples.

7. Unit user's guides.

The responsible organization for the User's Guide will be Commercial Contractor Company, Research & Development Department.

4.3 Installation Guide

The GENERIC System Installation Guide describes installing the GENERIC software on a Digital VAX 11/785 under VMS 4.3. The Guide describes a command procedure to load, compile, and link software units and provides instructions to install the software system. A diagnostic test procedure is provided to verify the system. The command procedures will:

1. Copy the GENERIC software to disk from tape.

2. Compile and link the GENERIC software.

3. Link libraries as needed.

4. Link the database management system.

5. Perform logical unit assignments as necessary.

The responsible individual for the Installation Guide is the Project Leader.

4.4 Test Results Summary

For all test cases the following will be supplied in the documentation:

1. The inputs and the outputs.

2. The modules and functions specifically being tested.

3. The expected results.

4. The actual results.

The responsible organization for the Test Results Summary will be the Testing Subcommittee.

4.5 Software Unit Documentation

Each software unit under the GENERIC project will be developed by the Software Development Groups, who are responsible for providing and reviewing the following documentation.

4.5.1 Preliminary Design Document

For each software unit, the Preliminary Design Document contains a description of:

1. Software unit objectives
2. Interfaces among hardware, software, and users
3. Major software functions
4. External files and databases
5. Design constraints and limitations
6. Reference documents
7. The design (to include descriptions of the data, the flow of information, and interfaces within the software)

4.5.2 Detailed Design Document

The Detailed Design Document contains:

1. Module descriptions, which include a description of the process, an interface description listing all data input and output from a module, and interfaces with other modules. The description should clearly describe the major tasks and processing that occurs within the module.
2. File structure descriptions, which include logical descriptions of the external files and the records.
3. Global data descriptions
4. A cross-reference between the requirements and the modules critical to implementation of the requirements.

5. Packaging and software transfer considerations, to include any high performance requirements or physical memory limitations which may cause modification in the design. Also, a description of the operating system characteristics necessary to understand the design.

4.5.3 Other Documents

In addition, for each software unit, the SDG's will develop a User's Guide, Installation Guide, and Test Results Summary document similar to those documents described above for the overall system.

4.6 Translator Software Units

Documentation for the various TRANSLATOR software units will comply with the documentation required under 4.5 for the GENERIC software units.

5.0 Standards, Practices, and Conventions

The GENERIC project will adhere to those particular standards and conventions for the project's software development life cycle extracted from *Sandia Software Guidelines*, Volume 3," Standards, Practices and Conventions." [Ref 2.2], and detailed in the GENERIC Project Plan [Ref 2.4]. Compliance with these standards is the responsibility of the project leader, and will be monitored and enforced by the project reviews and inspections.

6.0 Reviews and Inspections

Reviews and code inspections shall be performed at each major project milestone to ensure compliance with this SQAP and to aid early discovery and correction of software defects and errors prior to release of the software system.

Documentation of the results of the review or code inspection in a report which identifies all the deficiencies discovered along with a plan and schedule for corrective actions is required. Details on the reviews and inspections are provided in the GENERIC Project Plan [Ref 2.4], following the outline found in *Sandia Software Guidelines*, Volume 3," Standards, Practices and Conventions." [Ref 2.2]

The following is a list of the reviews and code inspections required:

1. Software Requirements Review and report.

2. Preliminary Design Review and report.

3. Detailed Design Review and report.

4. Software Test Plan Review and Functional Audit.

5. Code inspections and diagnostic reports.

The timing and conduct of these reviews is provided in the GENERIC project administrative plan. For each review and inspection, the results are documented in a report which identifies all deficiencies discovered during the review, and which provides a plan and schedule for corrective action.

7.0 Software Configuration Management

This section is addressed in detail in the project's Software Configuration Management Plan [ref 2.6], which explicitly follows the outline provided in *Sandia Software Guidelines*, Volume 4," Configuration Management."

8.0 Problem Reporting and Corrective Action

This section is addressed in detail in the project's Software Configuration Management Plan [ref 2.6].

9.0 Tools, Techniques, and Methodologies

During the design phase, the GENERIC project will employ the VAXBASED Structured Analysis Tool to create data flow diagrams to represent the data exchange relationship within the GENERIC software system and illustrate the dependency of processes on that data flow. The tool facilitates the definition and description of those processes. Following implementation, the Maintainability Analysis Tool will be used to audit the complexity of the code developed. These software tools are detailed in the GENERIC Project Plan [ref 2.4].

The AT&T Inspection Process, as described in Volume 3, *Sandia Software Guidelines*, will be used during all phases. The formal documentation required by this process will be collected as part of the quality assurance records.

10.0 Code Control

This section is addressed in detail in the project's Software Configuration Management Plan [ref 2.6].

11.0 Media Control

This section is not applicable to this plan. The GENERIC software will be made available to all interested parties; no environmental controls or offsite storage will be required.

Note: An explanation should be given as to why any of the thirteen IEEE sections do not apply.

12.0 Supplier Control

The only outside vendor software required in the GENERIC project is the commercially available Database Management System. This product has been in use at Sandia for several years and meets established technical requirements. No additional supplier control is applicable to this plan.

13.0 Records Collection, Maintenance, and Retention

Hard copies of GENERIC Project documents will be collected to provide reference data for Org 9800 development techniques, and to document the software quality techniques of the project. These records will be retained as outlined in the following table:

GENERIC Project Record:	*Retention Period:*
Inspection and review results	six months
Software verification reports	two years
Configuration control records	life of system
Test results (complete plus latest)	life of system
GENERIC system documentation	life of system
All other project documentation	one year

14.0 Testing Methodology

Testing will be the primary vehicle for ensuring that the code, when integrated into the software system, performs functionally in a manner consistent with software requirements, and that it does not perform unintended, undesirable functions under reasonable use conditions. In addition, software testing will be used to verify consistency of code with design specifications. Details on the testing procedures are given in the GENERIC Software Test

Note: This is not currently one of the thirteen required IEEE sections. Any such information should be added starting with section 14.0

Plan [ref 2.5]. The Sandia Software Guidelines, Volume 3 [ref 2.2] provides general guidelines for testing.

APPENDIX: Project GENERIC Glossary and Acronyms

- DBMS - database management system
- FILEWRITER - The GENERIC software unit that converts a vendor database into a GENERIC file.
- GENERIC software - The software units PARSER, KERNEL, and FILEWRITER.
- GENERIC System - The collection of software units designed to function together as the graphics file exchange system.
- KERNEL - The GENERIC software unit consisting of software procedures that will provide specialized access to the database.
- Lead Software Engineer - The software professional charged with the technical leadership and management of the GENERIC project.
- PARSER - The GENERIC software unit that loads a vendor file into the database management system.
- quality assurance – The planned and systematic actions necessary to provide adequate confidence that an item is fit for intended use.
- software – Computer programs, procedures, documentation, and data pertaining to the operation of a computer system.
- Software Development Groups (SDG) - The organizations responsible for the GENERIC software units.
- SQA - software quality assurance
- TRANSLATOR - The software procedures that convert the format generated by a specific graphics system to (or from) the GENERIC format.

Appendix B

References

AL84 Department of Energy/Albuquerque Operations Office
DOE/AL Quality Assurance (AL 5700.6B), Quality Engineering Division, April 24, 1984.

EPd87 Sandia Engineering Procedures
EP401511, draft February 1987.

IEE84 The Institute of Electrical and Electronics Engineers, Inc.
ANSI IEEE Standard for Software Quality Assurance Plans, IEEE Std 730-1984, New York, 1984.

IEE86 The Institute of Electrical and Electronic Engineers, Inc.
IEEE Guide for Software Quality Assurance Planning, IEEE Std 983-1986, New York, 1986.

QC85 Department of Energy/Albuquerque Operations
DOE/AL Quality Control Policy (QC-1), Quality Engineering Division, Apr 1985.

SQP86 Sandia National Laboratories
Quality Plan, Sandia National Laboratories, Albuquerque, NM, Apr 1986.

SSGv2 Sandia Software Guidelines
Volume 2, *Documentation,* SAND85-2345, Sandia National Laboratories, Albuquerque, NM, *expected printing* Dec 1988.

SSGv3 Sandia Software Guidelines
Volume 3, *Standards, Practices, and Conventions,* SAND85-2346, Sandia National Laboratories, Albuquerque, NM, Jul 1986.

SSGv4 Sandia Software Guidelines
Volume 4, *Configuration Management,* SAND85-2347, Sandia National Laboratories, Albuquerque, NM, *expected printing* Jun 1988.

SSGv5 Sandia Software Guidelines
Volume 5, *Tools, Techniques, and Methodologies,* SAND85-2348, Sandia National Laboratories, Albuquerque, NM, *expected printing* Jan 1988.

Appendix C

Glossary and Acronyms

Where possible, definitions in this glossary are taken from the *IEEE Standard Glossary of Software Engineering Terminology*, [IEE83g]. They are included here to provide a single-source document for the reader.

ANSI: American National Standards Institute

audit: An independent review for the purpose of assessing compliance with software requirements, specifications, baselines, standards, procedures, instructions, codes, and contractual requirements.

baseline: A specification or product that has been formally reviewed and agreed upon, and thereafter serves as the basis for further development; changed only through formal change control procedures.

change control: The process by which a change is proposed, evaluated, approved or rejected, scheduled, and tracked.

configuration control board: The authority responsible for evaluating and approving or disapproving proposed engineering changes, and ensuring implementation of the approved changes.

configuration management: The process of identifying and defining the items in a system, and controlling the release and change of those items throughout the system life cycle.

DOE: Department of Energy

IEEE: The Institute of Electrical and Electronic Engineers, Inc.

milestone: A scheduled event for which some project member or manager is held accountable and that is used to measure progress.

QA: quality assurance

quality assurance: A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements.

quality control: Inspections of documents and workproducts produced at each milestone, to demonstrate if the items are within specified criteria.

requirement: A condition or capability that must be met by a system or system component to satisfy a contract, specification, or other formally imposed document. The set of all requirements forms the basis for system development.

SQA: software quality assurance

SQAP: software quality assurance plan

SSG: Sandia Software Guidelines

software: Computer programs, procedures, rules, and associated documentation and data pertaining to the operation of a computer system.

software maintenance: Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment.

software professional: One who develops or maintains software for others.

software quality assurance: [see **quality assurance**]

software quality assurance plan: The methods a project will employ to assure the documents or workproducts produced and reviewed at each milestone are of high quality.

software reliability: The ability of a program to perform a required function under stated conditions for a stated period of time.

system: An integrated whole composed of diverse, interacting specialized structures and subfunctions.

validation: The process of evaluating software at the *end* of the software development process to ensure compliance with software requirements. [see also **verification**]

verification: The process of determining whether or not the products of a given *phase* of the software development cycle fulfill the requirements established during the previous phase. [see also **validation**]

WR: war reserve

Index

Note: Page numbers in **boldface** reference a subsection on the indexed term. Page numbers in *italics* reference a definition of the term.

access,
    physical 20
AL 5700.6B *2*
audit **6, 10**, 16, 29,*48*
    report 6

baseline 8, 9, *48*

change control 15, *48*
configuration
    control board 6,9,10,31-32,*48*
control,
    change 15, 48
    code 18
    media 18
    quality 6, 10, *48*
    supplier 18, 20

design
    information **15**
    review **15**
distributing
    copies 19
documentation, 19
    engineering **15**

environment **2**
evaluation
    of SQAP **27**
        form, content 28
        form, administration 29
execution
    of SQAP **26**

forms
    SQAP administration evaluation **29**
    SQAP content evaluation **28**

implementation **25**

management
    commitment 25
    participation 25
milestone *48*

project
    records 21
        collection 22
        maintenance 23
        retention 23

QC-1 *2*
quality assurance *48*
quality control **6,10**, *48*
quality operations division 10, 29
quality plan **2**

requirements *49*
review
    and audits 16
    design 15
    schedule 23

schedule,
    review 23
software *49*
    maintenance *49*
    professional *49*
    quality **1**
    quality assurance 5,*49*

division 3-6, 10,11,13,14,29,30
evaluation of plan **28**
example of plan **34**
execution of plan **27**
modification of plan **30**
plan *49*
program 2
reliability *49*
standards, practices, and conventions 10, 16
storage,
physical 19, 20
system *49*

tools, techniques, and methodologies 16

validation 8,10, *49*
verification 8,10, *49*

Distribution:
Sandia Internal:
123 H.L. Crass
1251 T.A. Borkey
1265 P.L. McAllister (2)
1410 P.J. Eicker
1411 R.C. Lennox (2)
1412 S.J. Weissman (2)
1413 Ronald Devries
1522 Bob Reuter
1523 J.H. Biffle
1531 J.M. McGlaun (2)
1556 P.C. Kaestner (2)
2113 J.A. Wisniewski (5)
2113 J.A. Hudson (2)
2124 Dan R. Royalty
2300 J.L. Wirth
2311 H.D. Pruett
2312 Betty P. Chao
2312 Dave D. Neidigk
2312 M.J. Smartt (2)
2314 Donald M. Small (2)
2335 T.J. Allard (2)
2336 C.R. Borgman (2)
2526 Julie Darling
2610 D.C. Jones
2612 D.M. Darsey
2614 A.R. Iacoletti
2614 Philip Campbell
2614 Betty Straba (100)
2620 K.O. Waibel
2621 Manny P. Ontiveros (2)
2621 Peggy Schlesinger (2)
2624 J.R. Schofield, Jr. (2)
2640 E.J. Theriot
2641 L.D. Buxton (2)
2642 P.A. Lemke (2)
2644 R.E. Jones (2)
2800 W.E. Alzheimer
2810 D.W. Doak
2811 T.R. Perea
2811 R.J. Harrison (2)
2811 Lilita Meirans (2)
2811 R.E. Parks (2)
2812 J.F. Jones, Jr. (2)
2812 T.F. Ezell (2)
2812 L.M. Grady (2)
2813 S.K. Fletcher (2)
2813 C.K. Miller (5)
2813 S.L.K. Rountree (5)
2814 P.A. Erickson
2814 M.A. Blackledge (10)
2820 G. Carli
2822 R.S. Kramer
2822 Susan Navarro (5)
2825 Jonathan Lee (2)
2826 A.J. Ahr (2)
2830 G.R. Urish
2830 Jerry D. Stauffer (2)
2833 Joe Lopez
2850 J.L. Tischhauser
2854 Steve Baca
2854 N.J. Nelson (2)
2854 S.C. Babb (5)
3151 R.L. Manhart
3538 B.H. VanDomelen
5100 H.W. Schmitt
5164 Don Schroeder
5164 J. Arlin Cooper
5164 M.W. Sharp (2)
5246 L.M. Desonier (2)
5253 R.L. Craft
5255 R.D. Halbgewachs
5255 P.W. Harris
5255 Bruce N. Malm (2)
5255 A.L. Yates (5)
5263 R.F. Davis (2)
5268 C.E. Olson

5268 M. Rodema Mosely
5268 M. Sagartz
6310 R.R. Richards (2)
6316 E.W. Shepherd (5)
6415 F.E. Haskin (2)
6415 L.T. Ritchie (2)
6416 Ginger Wilkinson (2)
6418 John E. Kelly
6418 S.W. Webb (2)
6440 D. Brosseau (2)
7111 James S. Phillips (2)
7200 C.H. Mauney
7201 C.A. Trauth (5)
7233 R.E. Smith
7233 R.M. Axline
7233 S.C. Billups
7233 S.E. Ohrt
7243 B.G. VanBlaricum
7251 Mary E. Prickett (2)
7251 W.P. Thomas
7252 D.P. Patrick (2)
7253 Guy E. Dahms (5)
7253 G.A. Gurule
7254 S.L. Sardalos (2)
7260 J.A. Hood
7262 Frank A. Ross
7262 D.G. Adams (2)
7262 R.B. Ronan (2)
7263 G.W. Mayes (2)
7483 Paul W. Plomp
7521 S.Y. Goldsmith (2)
7524 W.D. Swartz (2)
7525 David N. Harstad
7543 William L. Larson
8235 D.L. Crawford (2)
8274 R.E. Isler (2)
8343 Tim P. Tooman (2)
8474 J.N. Rogers (5)
9013 R.D. Summers (2)
9211 Gayle M. Connor (5)
9211 J.E. Lenberg
9211 Karen Weber
9212 Bill Jacklin (2)
9212 Janet Carkeet (2)
9213 Chuck Kyger
9221 D.H. Rountree (2)
9224 M.T. McCornack (2)
9224 Merry Peterson (5)
9224 J.C. Rowe (2)
9224 W.J. Slosarik (2)
9241 P.S. Hamilton (5)
9242 L.M. Ford (2)

3141 S.A. Landenberger (5)
3151 W.L. Garner (3)
3154-1 C. H. Dalin (28)
for DOE/OSTI
8024 P.W. Dean