NISTIR 4909

Software Quality Assurance: Documentation and Reviews

Dolores R. Wallace
Wendy W. Peng
Laura M. Ippolito

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 4909

Software Quality Assurance: Documentation and Reviews

Dolores R. Wallace
Wendy W. Peng
Laura M. Ippolito

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

September 1992

U.S. DEPARTMENT OF COMMERCE
Barbara Hackman Franklin, Secretary

TECHNOLOGY ADMINISTRATION
Robert M. White, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

ABSTRACT

This study examines the contents of a software quality assurance standard for nuclear applications. The study includes recommendations for the documentation of software systems. Background information on the standard, documentation, and the review process is provided. The report includes an analysis of the applicability, content, and omissions of the standard and compares it with a general software quality assurance standard produced by the Institute for Electrical and Electronics Engineers. Information is provided for the content of the different types of documentation. This report describes information for use in safety evaluation reviews. Many recommendations in this report are applicable for software quality assurance in general.

KEY WORDS

Documentation; Nuclear Power Plants; Project Management; Reviews; Safety Evaluation; Software Configuration Management; Software Design; Software Quality Assurance; Software Requirements; Software Safety; Software Verification and Validation; Test

EXECUTIVE SUMMARY

This study was funded by the United States Nuclear Regulatory Commission (NRC) under interagency agreement RES-91-003 between the NRC and the National Institute of Standards and Technology (NIST). As requested by NRC, this report presents a review of Part 2.7, "Quality Assurance Requirements of Computer Software for Nuclear Facility Applications," ASME NQA-2a-1990 [ASMENQA2], and a review plan for the analysis of software documentation. The results of this study are presented to NRC for consideration and are not endorsed by NRC.

While this study was conducted relative to the assurance of software for safety systems in nuclear power plants, the results and recommendations contained in this report are applicable to other critical software safety systems. This analysis of documentation and review processes resulted in identifying the issues and tasks involved in software quality assurance (SQA). It also revealed that because most standards are generic (i.e., do not address the needs of a specific industry), tailoring of standards to specific domains must be done to ensure a complete safety evaluation.

[ASMENQA2] is an SQA standard written by the American Society of Mechanical Engineers (ASME) specifically for the nuclear industry, based on two standards from the Institute for Electrical and Electronics Engineers (IEEE). These two standards are IEEE Std 1012-1986, for software verification and validation (SV&V) plans, and IEEE Std 730-1984 [IEEE7300], for SQA plans. [IEEE7300], the first of IEEE's SQA standards, was revolutionary in its time, but has since been upgraded. Today, as practitioners gain experience in software engineering and software quality, working groups for standards under revision are more carefully examining system-software relationships, requirements for software safety and computer security, differences in software test types, and the relationship between software development activities and software assurance activities.

The objectives of this study are to recommend improvements for [ASMENQA2], and to identify documentation practices and questions that will aid NRC reviewers in determining if a product should be licensed for use within nuclear power plants. In this review, the contents of [ASMENQA2] are compared against those of a revised version of the IEEE SQA standard, IEEE Std 730.1-1989.

Some issues this report addresses include: evaluations of software products based on predicate devices; documentation requirements that emphasize software safety and computer security; software properties that may affect the safety of the total system. This report concentrates on activities for the software lifecycle. It also identifies some issues that are of special interest when the software lifecycle is embedded within the system lifecycle. One of these issue concerns information from the system requirements phase that is essential for the software development phases. Another of these issues concerns the scope and definition of *software* verification and validation within system verification and validation (this report does not address system verification and validation).

This report provides recommended content for documentation of each software life cycle activity. The activities include not only the development phases from software requirements to software maintenance, but also project management, software configuration management (SCM), software verification and validation (including testing), and SQA.

NRC reviewers may wish to examine the results of formal reviews performed by the vendor. Such examinations show whether or not the vendor properly implemented SQA throughout the development process, and may reveal insights on how well the product has been built. This report provides the generic procedures for formal development reviews in an appendix, with detailed lists of questions for specific reviews.

[ASMENQA2] addresses activities for software lifecycle phases from software requirements to software maintenance, as well as some assurance activities, but does not necessarily require documentation or review of all these activities. Some SCM documentation is identified, but there is no requirement for an SCM plan. Project management and SQA elements such as metrics and training are not addressed. Software testing is identified as one activity when in reality software testing is performed at different levels with different objectives. [ASMENQA2] does not specify unique assurance requirements for software safety or computer security. Although [ASMENQA2]'s title indicates that it has been written specifically to provide SQA requirements for the nuclear industry, only one section addresses issues specific to nuclear facilities (SV&V section mentions (un)intended functions).

Although the required scope for this study includes only safety systems in nuclear power plants, the recommendations for documentation and reviews are presented in a broader context to ensure completeness. While the recommendations may be applied to systems of varying size, the purpose in expanding the scope is to provide NRC reviewers with a comprehensive understanding of what comprises a quality product. Thus, depending on the product under review, NRC may decide that not all the recommendations are appropriate. The recommendations address topical issues as guidance for NRC reviewers but are not intended as the only resource for details to be checked by the reviewers.

[ASMENQA2], its analysis, and all recommendations in this report are based on the waterfall lifecycle model. Readers should be aware that the recommendations may not be totally applicable when other lifecycle models, computer aided software engineering (CASE) tools, modern methodologies, and state-of-the-art concepts for documentation (e.g., hypertext) are used.

TABLE OF CONTENTS

TABLES

1. OVERVIEW

This study was funded by the United States Nuclear Regulatory Commission (NRC) under interagency agreement RES-91-003 between the NRC and the National Institute of Standards and Technology (NIST). As requested by NRC, this report presents a review of Part 2.7, "Quality Assurance Requirements of Computer Software for Nuclear Facility Applications," ASME NQA-2a-1990 [ASMENQA2][1], and a review plan for the analysis of software documentation. The results of this study are presented to NRC for consideration and are not endorsed by NRC.

While this study was conducted relative to the assurance of software for safety systems in nuclear power plants, the results and recommendations contained in this report are applicable to other critical software safety systems. This analysis of documentation and review processes resulted in identifying the issues and tasks involved in software quality assurance (SQA). It also revealed that because most standards are generic (i.e., do not address the needs of a specific industry), tailoring of standards to specific domains must be done to ensure a complete safety evaluation.

This review is intended to provide NRC reviewers with guidelines for evaluating software used in nuclear power plant safety systems. These evaluations are based on software documentation. Utilities submit vendor products to NRC for approval at the end of product development, and sometimes during development.[2] NRC may conduct audits during the development process. NRC reviewers evaluate these products based on the documentation, reviews, and reports supplied by the vendor. They check for compliance with product plans, safety requirements, and with recommended standards and guidelines for software development and assurance. NRC may be asked to review other types of software products, which may be only one component of a system, or NRC may examine an entire system, consisting of software and other components. The recommendations in this report apply only to the software documentation aspects of the safety evaluation.

The purpose of this report is to aid the NRC reviewers during safety evaluations. However, this report also includes an appendix which discusses formal reviews conducted at milestones (e.g., at completion of a software lifecycle phase or when necessary to discuss a serious concern). The formal review is conducted by the purchaser and developers of software; this type of review is also appropriate within a utility when it develops the software. Other types of reviews can also be conducted by individuals who prepare reports on products for discussion at the formal review meetings or who, as in the case of NRC reviewers during a safety evaluation, examine the product, its technical documentation and project documentation.

Although the frame of reference for this study includes only safety systems in nuclear power plants, this report includes recommendations for documentation and reviews in a broader context to ensure completeness. While some software safety systems may be small, other computer

[1]Complete references for documents referred to by acronyms are in Section 7.

[2]A utility may also develop a product. The term vendor will be used to refer to vendors or utilities.

systems in the nuclear industry may be larger, may have more complexity and interfaces, and may access or generate large databases. NRC reviewers should have a comprehensive understanding of a complete quality program, regardless of the size and characteristics of the product under review. Depending on the particular software product, NRC reviewers may decide that not all the recommendations in this report are appropriate or may add additional safety-related review items to improve the coverage of safety issues.

Section 2 of this report includes background information on the uses of [ASMENQA2], documentation, and reviews. Section 3 details the contents of [ASMENQA2] and cites its omissions. Section 4 identifies the necessary types of documents, and provides suggestions for their content. Section 5 provides recommendations for NRC reviewers conducting safety evaluations. Appendix A contains definitions of quality attributes. Finally, Appendix B provides information on the formal review process (which may assist NRC during safety evaluations).

1.1. Acronyms

Acronyms used in this report (other than those used to refer to documents) are listed below.

O&M	Operation and Maintenance
PMP	Project Management Plan
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SDD	Software Design Description
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
SV&V	Software Verification and Validation
SVVP	Software Verification and Validation Plan
SVVR	Software Verification and Validation Report

2. BACKGROUND INFORMATION

[ASMENQA2] may be used by the vendor to ensure quality in the product, or by the NRC reviewer in safety evaluations. When [ASMENQA2] is the designated governing document, the vendor is not obligated to perform any SQA activity not specifically identified, even if recognized in other standards or documented practices.

The vendor reviews documentation from a proactive perspective and must adjust processes and software products based on review findings. For the vendor, an SQA plan (SQAP) must provide significant detail, or reference detailed requirements. For consistency within the nuclear industry, the SQAP should be based on a rigorous standard. If NRC is monitoring development of a product, then NRC roles are also proactive.

When NRC reviewers evaluate a completed product for approval for use within a nuclear power plant, the NRC reviewer is reactive. The reviewer must understand the development and assurance processes that were applied to the product. To do this, the reviewer examines documentation of the processes to seek evidence that the vendor has fulfilled all the applicable requirements for software in safety systems for nuclear power plants. The reviewer will make this determination based in part on standards and practices adopted or recommended for use by the nuclear industry.

[ASMENQA2] addresses requirements for lifecycle activities, documentation, reviews, and several SQA activities such as records management and procurement procedures. Vendors may use [ASMENQA2] to plan the software quality program. For this reason, this report identifies additional practices, documentation and reviews that may be applicable.

NRC reviewers should be aware of the total spectrum of issues for software quality. Thus, this report provides general recommendations for documentation and review. In some instances, these recommendations may not be applicable to software safety systems in nuclear power plants. NRC should recommend that vendors justify why a particular document, specific contents of a document, or an activity, are omitted. The justification may be a simple statement (e.g., database documentation is not provided because there is no database).

Documentation serves as a communication medium informing all involved persons of their responsibilities and the technical specifications related to the product. Documentation also serves as a communication medium for those who build the product, monitor the project, and review the products and processes of the project. Reviewers during development will examine product documentation to learn how the product is being built, while the reviewer at final product evaluation examines how the product was built. The reviewer during development examines plans and interim products to determine whether or not the planned assurance and development activities are sufficient and the evolving product is meeting its assurance criteria. The reviewer at final product evaluation will attempt to verify that appropriate engineering practices were used, assurance activities were properly applied, problems found during development and validation have been corrected, and the resulting product meets its specifications.

This report provides recommendations for reviews by NRC and provides discussion on reviews by the vendor. In vendor reviews, the primary purpose is to make immediate decisions based on the results of the review. NRC is not present for these reviews. However, in response to a utility's request, NRC may attend reviews or examine project documentation at some point during development; in this case NRC's role is proactive. Descriptions of these reviews are included in this report to aid NRC reviewers' understanding of procedures that should have been performed during development. For the safety review by NRC, the procedures and types of questions are different than those of the formal reviews conducted by the vendors. However, understanding and the results of the development (formal) reviews may be important aids to NRC reviewers.

[ASMENQA2], its analysis, and all recommendations in this report are based on the waterfall lifecycle model. Readers should be aware that the recommendations in this report may not be totally applicable when other lifecycle models, CASE tools, modern development methodologies, and state-of-the-art concepts for documentation (e.g., hypertext) are used.

3. CONTENTS OF ASME'S QUALITY ASSURANCE STANDARD

[ASMENQA2] has been tailored from [IEEE1012] and [IEEE7300][3], standards on software verification and validation plans (SVVPs) and SQAPs, respectively. [ASMENQA2]'s title indicates that it has been written specifically to provide SQA requirements for developers in the nuclear industry. However, only one section addresses issues specific to nuclear facilities (SV&V section mentions (un)intended functions). Table 3-1 contains recommended SQA activities and documentation and a comparison of [ASMENQA2] and [IEEE7301] relative to these recommendations. [IEEE7301], the most current version of the IEEE SQAP standard, is used because it is more complete than previous versions. While Table 3-1 indicates that both standards address most topics, it does not show the extent of the requirements for each topic. This section provides an overview of [ASMENQA2]'s general contents.

[ASMENQA2] states that while any software lifecycle model may be used, the model must encompass the activities associated with the model of [IEEE1012] for requirements, design, implementation, test, installation and checkout, and operation and maintenance (O&M). [ASMENQA2] addresses other lifecycle processes, which are referred to as assurance activities in [NIST204, NUREG5930]. One feature of [ASMENQA2] that may be confusing is the fact that it runs through the lifecycle three times. First, it describes the activities of each lifecycle phase. It then lists the documentation for each phase. Finally, it addresses the reviews of each lifecycle phase.

[ASMENQA2] identifies lifecycle activities and some assurance activities but does not necessarily require documentation of these activities. [IEEE7301] identifies basic SQA documentation requirements without requiring lifecycle activities. However, since [IEEE7301] is a standard describing the content of SQAPs, it is acceptable that it does not explicitly describe SQA activities. [ASMENQA2] cites some documentation requirements from [IEEE7301], identifies lifecycle activities, and specifies reviews on some development products. It does not specify reviews of all lifecycle activities.

If a standard requires an activity, the standard should contain requirements for the documentation, review, and monitoring of the activity. The inclusion of statements such as the following would be useful: each lifecycle activity shall be documented; reports shall be prepared on each activity's implementation; an SQA review shall verify that the lifecycle activity plan is acceptable and was accomplished (or if not, why not); tracking mechanisms shall provide information to project management and reviewers.

[ASMENQA2] and the IEEE standards are intended for specific projects, and therefore do not address a vendor's quality management program. A plan based on [ASMENQA2] may reference the vendor's general quality management program (e.g., one based on [ISO9000]).

[3]While IEEE 730-1984 [IEEE7300] has been replaced by IEEE 730.1-1989 [IEEE7301], the authors believe that the earlier version of the SQAP standard guided development of [ASMENQA2].

Requirements	[ASMENQA2]	[IEEE7301]
Assurance Activities		
■ SCM	yes	N/A
■ SV&V	yes	N/A
SQA Plan Elements		
■ organization for quality responsibilities	yes	yes
■ identification of applicable software products	yes	yes
■ standards, practices, conventions	yes	yes
■ metrics	no	yes
■ reviews	yes	yes
■ audits	no	yes
■ documentation	yes	yes
■ error reporting and corrective action	yes	yes
■ access control	yes	no
■ procurement/supplier	yes	yes
■ records management	yes	yes
■ training	no	yes
■ tools, techniques, methodologies	yes	yes
■ risk management	no	yes
Required Documentation		
■ Project Management Plan	no	no
■ SQA Plan	yes	yes
■ SCM Plan	no	yes
■ Configuration Identification Documents	no	yes
■ Configuration Change Control Documents	yes	yes
■ Configuration Status Accounting Documents	yes	yes
■ SCM Audits and Reviews Documents	no	no
■ SV&V Plan	yes	yes
■ SV&V Reports (includes Test Reports)	yes	yes
■ Software Requirements Specification	yes	yes
■ User's Manual	yes	yes
■ Software Design Description	yes	yes
■ Source Code Documents	no	Stds & Procedures Manual
■ Installation and Checkout Document	no	no
■ O&M Manual	no	no
■ Maintenance Documents	yes	no
Required Reviews		
■ Management Reviews	none	SVVP, SCMP
■ Software Requirements Review	yes	yes
■ Software Design Review	yes	yes, 2 reviews
■ Source Code Review	no	no
■ Test Report Review	no	no
■ Development Documentation Review	yes	no
■ Installation & Checkout Review	no	no

Table 3-1. Comparison of Requirements

Project Management

Project management is clearly indicated as a basis for software development maturity [SEI91], but neither [ASMENQA2] nor [IEEE7301] require a project management plan (PMP). The argument for its omission may have been that the PMP specifies SQA activities; thus an SQAP would not include requirements for a PMP. However, an SQA standard, especially one that has already been tailored for a particular use, should be complete in identifying all the documentation and activities that comprise a quality project. When NRC reviews the products during development, the NRC reviewers should review the PMP to assess potential capability of the vendor to satisfy the requirements, and ensure that the plans for development were reasonable to begin with. NRC reviewers at final product evaluation should also examine the project management documentation to ensure that the project has been well-managed and that all essential elements of the project have been addressed.

Software Quality Assurance

[ASMENQA2] does not address measurement, the use of metrics for measuring quality. Measurement of the software development process and product is valuable to the developing organization. NRC reviewers should be interested in the vendor's reasons for changing processes and technical specifications. The SQAP should identify what measurements are to be taken, when they are to be examined, by whom, their purpose (e.g., to determine if there is a problem, to evaluate effect of change in development process, to examine effectiveness of testing, inspection or other analysis techniques) and who has authority to make changes in any of the processes.

Information on measurement in the SQAP is helpful to NRC reviewers. For example, periodic checks of test results (e.g., number of errors found) may reveal an error rate outside the control limits. A reviewer may discover that this is due to the fact that inspections were eliminated when the project fell behind schedule. When NRC is involved during development, actions can be recommended to correct the situation. At final product review, if the NRC reviewers have access to problem report data, they can question the reason for the higher error rate and may use this information to make judgments about the fitness of the final product.

[ASMENQA2] does not address risk management. Since the nuclear community in general must perform system hazard analysis this may be acceptable. However, a citation for risk management in a "software" standard may assure that software risks, or potential hazards, are to be identified from the system hazard analysis reports, and that these risk areas receive extra attention from the SQA activities.

[ASMENQA2] does not specify required audits. The audits identified in [IEEE7301] are conducted during development, in a contractual environment. NRC has the option to conduct audits during the development process, and at final safety evaluation. At the audits, many of the typical audit questions will be asked at this time, but from a reactive perspective.

[ASMENQA2] does not address training staff to conduct SQA activities. Again, [ASMENQA2] is intended to be a plan for a specific project; training may be covered in the vendor organization's general quality management program. For completeness, however, the SQAP may cite any special training for the project. With this knowledge from the plan, NRC reviewers can form expectations regarding the product quality.

Software Configuration Management

[ASMENQA2] identifies several major software configuration management (SCM) activities and requires documentation of some of these activities. For configuration change control, [ASMENQA2] states that "changes to software shall be formally documented," and for configuration status accounting, "the information that is needed to manage a configuration shall be documented." However, there is no requirement for an SCM plan (SCMP) or other SCM documents (e.g., documentation of SCM audits and reviews). For any project involving more than one person, an SCMP is needed to communicate how SCM will be implemented. The absence of SCM documentation can become significant when a reviewer at final product evaluation tries to identify whether the software products under evaluation are actually those products which were verified and validated.

Software Verification and Validation

[ASMENQA2]'s documentation requirements for SV&V include most of the requirements from [IEEE1012], but does not describe them in detail. SV&V documents such as the SVVP and SV&V reports are not specifically mentioned, although some contents of these documents are identified (e.g., description of SV&V tasks, criteria for accomplishing these tasks, results of execution of SV&V activities, test results). Requirements for the content of SV&V documentation should be more detailed in order to ensure that the intent of the requirements are fulfilled. SV&V must be planned, managed, and documented even for small projects.

Software Test

Testing is part of the SV&V process. Preparation and planning for the execution of testing occurs throughout the lifecycle, while the execution itself occurs more or less in a phase. Although [IEEE1012] provides detailed requirements for each major category of software testing (component, integration, system, acceptance), [ASMENQA2] treats all testing generically. This can be a problem because each of the major test types has different objectives. Reviewers during development need to verify that the objectives are appropriate and sufficient for product assurance, and reviewers at final product evaluation need to verify that those objectives have been met.

Acceptance test, which is the final set of validation activities for a product, is not addressed by [ASMENQA2]. In some environments, acceptance tests are planned and performed by the customer. In other environments, the vendor, with customer approval, plans and conducts acceptance tests. NRC should recommend to the nuclear industry that documentation for acceptance test (e.g., plans, procedures, results) should be submitted in the package for NRC review of safety features.

Software Requirements

[ASMENQA2] has tailored [IEEE1012] for a specific use. A major weakness of [IEEE1012] and [IEEE7301] is their lack of emphasis on the relationship of software to the system in which it resides and to software safety and computer security requirements.[4] This report attempts to address these concerns.

[ASMENQA2] is directed at the software only. Software safety systems may be embedded in hardware, and the typical software requirement specification (SRS) may not define how the software fits into the total system. This information should be provided in the SRS. While it is acceptable that the concept phase of [IEEE1012] is not addressed in [ASMENQA2], some information may need to be provided that usually is discussed before deciding to build a product (in the concept phase). For example, some functions of a software safety system may have previously been performed by non-software components. The rationale for making the transition from hardware to software could be important in helping NRC reviewers assess the completeness of the software requirements and whether they have addressed all potential hazards impacting software.

[ASMENQA2] identifies the minimum content of the SRS which includes a description of functionality, performance, external interfaces, design constraints, and quality attributes. Requirements for safety, computer security, and user documentation are missing. Requirements specific to software safety need to be identified (e.g., trace to test cases, trace through the system documentation) to assure adequate attention to these requirements throughout assurance activities.

Software Design and Implementation

[ASMENQA2] specifies general design requirements; however, no consideration is given to security or safety. [ASMENQA2] does not address source code documentation.

Installation and Checkout

[ASMENQA2] requires documentation of the approval of the software for operational use.

Operation and Maintenance

[ASMENQA2] requires that all modifications made to the software during the O&M phase should be documented.

[4] The current working group for the revision of IEEE1012 plans to accommodate these issues.

Development Documentation Review

[ASMENQA2] requires review of development documentation, but does not define what development documentation includes. A vendor can easily claim compliance to this requirement by reviewing almost any documentation, but the intent of the requirement may not be fulfilled. Section 4 discusses development documentation.

4. DOCUMENTATION RECOMMENDATIONS

Project documentation may include many kinds of documents (e.g., plans, task reports, development products, problem reports, phase summary reports). Project size, criticality (i.e., the severity of the consequence of failure of the system), and complexity are some features that may affect the amount of documentation a project should need. For example, the design documentation may consist of a single document describing both the system architecture and the detailed modules or it may consist of separate documents for the architecture and subsystems. The purpose of this section is not to specify how many documents should be required. Rather, this section identifies the information content needed for any project and the timeliness of requirements so that the information can be used by the vendor, the utility, and the NRC reviewers. Because the NRC reviewers cannot determine the characteristics of the software product without substantial technical specifications, project plans, and reports, NRC should specify the technical products of the vendor that the utility must provide NRC.

This report expands upon the documentation requirements in [ASMENQA2]. While minimum content requirements are provided for several documents (which include the [ASMENQA2] requirements), some documents may be presented together as a single document. Each of the planning documents may be kept simple and short, but the basic information identified in the recommendations should be addressed. Issues not addressed include the mechanics of document preparation, the medium of the documents (e.g., paper, disk), and requirements for modifying or updating documents. The impact of modern methodologies (e.g., information engineering) on documentation is also not addressed.

4.1. Project Management Documentation

PMP's should include project organization, methods, tools, reporting, assurance activities, cost and scheduling estimates. Some of this information may be included in the SQAP. The vendor may have a generic PMP but every project will have some specific information which must be provided. The following recommendations for a PMP are based on [IEEE1058].

PROJECT MANAGEMENT PLAN

- Introduction.
  - Project Overview. Summary of the project and its goals, the product, major work activities, major work products, major milestones, required resources, and master schedule and budget. Statement of requirements for assurance (e.g., SQAP, SCMP, SVVP).
  - Project Deliverables. Description of all items used by the developer and items delivered ultimately to the customer.

- Evolution. Procedures for reviewing and producing updates to the PMP.

- References, Definitions, and Acronyms.

■ Project Organization.

- Process Model. Relationships between project activities. Specification of timing of major milestones, baselines, reviews, work products, project deliverables, and sign-offs. Definition of project initiation and termination activities.

- Organization Structure. Description of the internal management structure including authority, responsibility, and communication.

- Organizational Boundaries and Interfaces. Project boundaries with other entities (customer, subcontractors, other system components). Interfaces of SCM, SQA and SV&V.

- Project Responsibilities. Description of major project activities and who has responsibility for each activity.

■ Managerial Process.

- Management Objectives and Priorities. The philosophy, goals, and priorities for management activities such as reporting mechanisms; priorities among requirements, schedule, and budget; risk management procedures; and, procedures relating to existing software.

- Assumptions, Dependencies, and Constraints. Assumptions the project is based upon, events on which the project is dependent, and constraints the project is to be conducted under.

- Risk Management. Project risk factors including contractual, technological (e.g., ability of technology to support the safety requirements), size/ complexity/ criticality, personnel acquisition and retention, and customer acceptance risks. Mechanisms for tracking the risk factors and implementing contingency plans.

- Monitoring and Controlling Mechanisms. Tools and techniques used in monitoring and controlling adherence to the PMP including reporting mechanisms, report format, information flows, and review and audit mechanisms.

- Staffing Plan. Types and numbers of personnel assigned, required skill levels, and training necessary.

- Technical Process.

  - Methods, Tools and Techniques. Methods, tools, and techniques used to conduct project activities. References to standards, policies and procedures used.

  - Software Documentation (may be included here or in SQAP). The documentation requirements, milestones, baselines, reviews, and sign-offs.

  - Project Support Functions. Reference to (or include here) the SCMP, SQAP, and SVVP. Responsibilities, resource requirements, schedules, and budget for the SCM, SQA, and software verification and validation (SV&V).

- Work Packages, Schedule and Budget.

  - Work Packages. Description of the work packages (i.e., task grouping) for the project activities.

  - Dependencies. Interdependencies among work packages and dependencies on external events.

  - Resource Requirements. Estimates of project resources including numbers and types of personnel, computer time, support software, computer hardware, office and laboratory facilities, travel and maintenance requirements.

  - Budget and Resource Allocation.

  - Schedule.

- Additional Components.

  Reference to additional items such as subcontractor management, security for the project, independent V&V, training, facilities, installation, or product maintenance plans.

4.2. Software Quality Assurance Documentation

The recommendations for the SQAP are listed in Table 3-1 and are discussed in section 3.

4.3. Software Configuration Management Documentation

There are several SCM documents that should be produced. The SCMP should describe how the SCM activities (configuration identification, configuration control, status accounting) will be conducted and documented. The other documents report on the SCM activities. The SCMP may exist as a generic plan which can be referenced by any project. Even on a small project it is easy to lose track of the relationship between tested software modules and modules in which software units have been changed. The following recommendations for an SCMP are based on [IEEE828]. [IEEE828] is used not only because of its merits but also because it is serving as the base document for an international standard in ISO/IEC JTC1 SC7 on Software Engineering.

SOFTWARE CONFIGURATION MANAGEMENT PLAN

- Introduction. Overview of the SCM process, the purpose and scope of the SCMP.
- Management. Procedures for establishing SCM organization, and for assigning tasks/responsibilities to all units in the organization.
- SCM Activities. Includes configuration identification, configuration control, configuration status accounting and report, and audits/reviews. Include in the description how these activities are to be implemented, which units are responsible for which activities, etc.
- Tools, Techniques, and Methodologies. Tools and methodologies to be used for implementing SCM activities.
- Records Collection and Retention. Procedures for identifying, assembling, filing, storing, maintaining, and disposing of SCM documentation.
- SCMP Maintenance. How the SCMP will be reviewed, updated, or revised while in use.

The following recommendations for the other SCM documents are based on [IEEE828].

CONFIGURATION IDENTIFICATION DOCUMENTS. Items that comprise a baseline and the procedures for identification of configuration items.

CONFIGURATION CHANGE CONTROL DOCUMENTS. Changes to the software. A description of the change, the rationale for the change, and the identification of affected baselines.

CONFIGURATION STATUS ACCOUNTING DOCUMENTS. Information that is needed to manage a configuration item including the following information: status of specifications; status of proposed changes; status of product versions or revisions; reports of approved changes; and, reports of the implementation of installed updates or releases.

SOFTWARE CONFIGURATION MANAGEMENT AUDITS AND REVIEWS DOCUMENTS.

4.4. Software Verification and Validation Documentation

For large or long term projects, the SVVP and the detailed test documentation should be separate documents. For small projects, while the initial planning and SVVP should be completed during the requirements phase, additional information regarding the SV&V tasks, test designs, case, and procedures may be added to the SVVP later. There are several types of SV&V documents that should be produced. The following recommendations for an SVVP are based on [IEEE1012].

SOFTWARE VERIFICATION AND VALIDATION PLAN

- Objectives. The objectives of performing SV&V should be stated. Sometimes thorough SV&V on every software requirement, design feature, and line of code may not be possible and specific objectives must be selected (e.g., to show that critical performance objectives have been met; to examine software interfaces to the system, especially during failures external to the software).

- Management Activities. Review of V&V Plan, review of SV&V results, monitoring of SV&V activities, policies for iteration of tasks, etc.

- Schedule of SV&V Tasks. Includes milestones and completion dates.

- Organizations. Organizations involved in the SV&V activities and specific responsibilities.

- Communication. Communication procedures between design team members, code team members, SV&V team members, and others involved in the project.

- Summary of Resources. Staffing, tools, facilities, and finances needed to perform SV&V tasks, etc.

- Facilities. Planning of facilities, especially tools for test activities (e.g., simulation, test case generation) that require acquisition, staff training, and test. Planning and providing for test facilities may take as long as software development itself, so adequate time should be allotted. Delayed facility planning may result in inadequate test strategies and execution.

- Testing. Distinction among the different types of tests, their objectives, and documentation.

- Lifecycle SV&V. Detailed descriptions of SV&V tasks throughout lifecycle. Identifies the following:

    - SV&V tasks and objectives for each lifecycle phase (e.g., the specific safety objectives as a focus of each task).

    - Methods and criteria used in performing the SV&V tasks.

    - Inputs and outputs required for each task.

    - Schedule for SV&V tasks.

    - Resources for performing SV&V tasks.

    - Assumptions concerning schedule, resources, or approach.

    - Responsibilities for performing the tasks.

- SV&V Reporting. Description of how results of implementing the SVVP will be documented.

- Anomaly Reporting and Resolution. Description of method for reporting and resolving anomalies, and tracking anomaly resolution.

- Policy for Reiterating a SV&V Task. Description of criteria used to determine the extent to which a SV&V task should be reperformed when input is changed.

- Policy for Deviating from the Approved Plan. A good reviewer during final product evaluation may pick up some differences between planned and actual activities and ask many questions about the deviation. If procedures were in place and were followed, vendor may be able to demonstrate that the deviation did not detract from quality of the software, or its SV&V.

- Configuration Control Procedures. Configuration management procedures on the documentation examined by SV&V tasks; if this information is included only in a SCMP, then reference to the SCMP should appear in the SVVP.

The following recommendations for other SV&V documents are based on [IEEE1012].

TASK REPORTS. Report on the status, interim results, or final results of a single well-defined task performed during a lifecycle phase. May include information about the relative efficiencies and difficulties of certain tasks, which may lead to adjustments in the development process, such as choice of methodologies or tools, reallocation of SV&V resources, alteration of schedules, etc.

SOFTWARE VERIFICATION AND VALIDATION PHASE SUMMARY REPORTS. Summary of interim or final results of all SV&V tasks performed during a lifecycle phase. Should be issued for each lifecycle phase. If SV&V tasks are performed by different organizations, the phase summary report will help to consolidate the evaluations of the different organizations.

ANOMALY REPORTS. The location, impact, cause, criticality, and possible recommendations for each anomaly. Should be generated throughout the lifecycle. Procedures for reporting anomalies might include using key words to characterize each anomaly, severity scales to assess its significance, and procedures for reviewing, approving, and dispatching of the reports.

SOFTWARE VERIFICATION AND VALIDATION FINAL REPORT. A summary of all SV&V tasks, summary of task results, summary of anomalies and resolution, assessment of overall software quality, and recommendations. This summary may be issued at the end of Installation and Checkout Phase or at conclusion of SV&V effort. It can be used as the baseline for maintenance, to help improve the development process for the next project, and to call attention to outstanding unresolved anomalies.

4.4.1. Test Documentation

Although test plans for component, integration, and system testing are usually separate documents, for small projects they may be presented together in one document. In these cases, the document should contain separate, complete sections for each type of testing, since the objectives and test strategies for each type of testing will still differ. If more than one organization is responsible for testing (e.g., developer performs component and integration tests, but an independent agent performs system tests), then separate documentation should be maintained. Such an arrangement should be specified in the SVVP. The following recommendations for test documentation are based on [NASA2100], [DOD2167A], and [FUJII].

TEST PLANS. Includes: objectives and requirements; basic strategy (e.g., how each type of testing will satisfy specific objectives; an example may be to state what design modules will undergo control flow analysis or data flow analysis); location and schedule of briefings, formal testing, and debriefings; facilities/tools; staff/resources; tentative schedule; review of test cases and procedures; configuration management, procedures for releasing test results; data reduction and analysis; and, review of test results.

TEST DESIGNS/CASES. Includes: objectives of classes of tests (e.g., logic tests, coverage, boundary tests); traceability to requirements; constraints; initialization; termination; inputs; expected outputs; criteria for evaluating results; and, interfaces exercised.

TEST PROCEDURES. Description of all test execution procedures including: equipment preparation; support software preparation; and, detailed test procedures (test scenario description; step by step instructions to perform test; actions required by equipment; action to perform in case of error; data reduction/analysis procedures).

TEST REPORT. Summarizes test execution and results. Includes: test records (date, time, location of test; test team members, witnesses; problems encountered, recovery procedures attempted, outcome of attempt); test results; and, evaluation and recommendations.

4.5. Requirements Documentation

4.5.1. Software Requirements Documentation

There may be functionality in the system whose sole purpose is to prevent a safety hazard from occurring. Such functions need special attention throughout the software development, especially in its documentation. The same holds for computer security requirements; if computer security is not an issue for a product, the SRS should include a statement to that effect. In an SRS that requires a quality attribute, a quantified description of that attribute should be included in the SRS; for characteristics of quality attributes see [SOFTENG]. The following recommendations for an SRS are based on [NASA2100], [ASMENQA2], NIST180, [IEEEP1059], [IEC880], and [IEEE830].

SOFTWARE REQUIREMENTS SPECIFICATION

The purpose of the SRS is to provide a description of the software. However, the SRS should also describe the relationship of the software component with the rest of the system. The description may include a graphical representation and an overview description of the services the software provides for the system. However, this information is not sufficient for any reviewer to judge the capability of the software to fulfill the overall system requirements.

- Functionality. Functions the software is to perform, and how inputs are transformed into outputs.
  - Functions. Description of all operations, equations, mathematical algorithms, logical operations, etc. Safety and security functions should be flagged.
  - Inputs. Definition of inputs: specify sources of inputs, types, formats, units of measure, timing, and ranges of valid inputs, including accuracies and tolerances, disposition of illegal values, error messages.
  - Outputs. Definition of outputs: specify destinations of outputs, units of measure, timing, range of valid outputs, including accuracies and tolerances, disposition of illegal values, error messages.
  - Data Handling. Requirements for database or dictionary.
- Performance. Specification of each performance requirement in testable and quantitative terms. The following should be addressed:
  - Timing and sizing requirements.
  - Sequence and timing of events, including user interaction tolerances.
  - Throughput and capacity requirements.
  - Error detection, isolation, and recovery requirements for data and processes.
  - Fail-safe requirements.

- Quality Attributes. Non-time-related requirements that software must meet such as usability, efficiency, reliability, maintainability, and portability.

- Design/Implementation Constraints. Constraints imposed externally (e.g., technical standards, marketing department), hardware limitations (e.g., use of a specific compiler, limits on memory), and by project requirements (e.g., safety features).

- External Requirements.

  - Interfaces (to other systems within or outside nuclear power plant). User, hardware, software, and communications interfaces. The purpose, requirements (e.g., performance, safety, security), and implementation constraints for each interface. For example, requirements for user interface may include screen descriptions and report descriptions.

  - Environment. Operating conditions (e.g., plant commissioning, refueling).

- Safety Requirements.

  - Identification of the stable modes and error conditions of the software.

  - Software requirements identified from hazard analysis.

  - User Interactions (e.g., requirements for signals that indicate safety problems and for information that notifies user of the status of other safety functions).

- Security Requirements. Includes access limitations to the system (e.g., log-on procedures and passwords), data protection and recovery methods. These requirements should be prioritized. Security issues may also be identified by hazard analyses.

- User Documentation. Description of requirements for content of user documentation.

- Man-Machine Specifications. Description of requirements for appropriate and timely signals between system and operator, automatic system actions to prevent danger or damage, etc.

4.5.2. User Documentation

[ASMENQA2] identifies the minimum content for user documentation including a description of the following: user interaction with the software, training, input and output specifications and format, system limitations, anticipated errors and user response, and user and maintenance support. However, requirements concerning security and safety (e.g., threats, how to override problems, how to respond to warning signals) are not specifically addressed. The following recommendations for user documentation are based on [NASA2100], [ASMENQA2], and [DOD2167A].

USER'S MANUAL

- Overview of Purpose and Functions. The purpose and main capabilities of the software, and its overall operation in terms of functions, options, and restrictions and limitations.

- Installation and Initialization[5]. The procedures for installing, tailoring, and initiating the software including the following:

    - Equipment setup.

    - Power-on and power-off.

    - Bootstrap and load.

    - Initiation commands.

    - Initialization of files, variables, or other data.

    - Tailoring, reconfiguration, adaptation.

- Restart. Procedures for restarting software after initialization failures, abnormal termination.

- Termination. Procedures for terminating operation, and determining whether normal termination has occurred.

- Recovery Steps. Recovery procedures in terms of: check points; collection of failure data; restoring files; and, restoring devices to operational mode.

- Inputs.

    - System inputs. Description of system inputs to the software that may occur while the software is in use and may affect the software's interface with the user (e.g., inputs from a remote sensor). Include applicable attributes of the input such as format, frequency, effect on the software state or mode, allowable range, and units of measure.

[5]For some systems, this section is included in the installation document, if installation is to be performed by the vendor, rather than the user.

  - User inputs. Description of user inputs including commands, data, and option selection.

- Outputs. Description of expected outputs of the software including results of functions, and error messages.

- Potential Hazards. Description of user response to unexpected events or problems that could cause serious hazards if user does not respond; description of activities that could cause the need for user response or override.

- Functions and their Operations. Description of each function in terms of: purpose of function; step-by-step procedures for execution; and, related functions.

- Error Messages. Error messages output by the software, the meaning of each error message, and the corresponding corrective actions to be taken.

4.6. Design and Implementation Documentation

In general, design rules will also be implemented by the source code (e.g., the modularity features and complexity of the design should transfer to the source code). It is important to ensure that the vendor's plans for modularity and complexity are stated in the design documentation and the modularity and complexity of the design are not lost during implementation. This report addresses design and implementation/source code in two separate documents.

4.6.1. Design Documentation

The following recommendations for a design document are based on [NASA2100], [DOD2167A], [APPGUIDE], P1059, and [ASMENQA2].

SOFTWARE DESIGN DESCRIPTION

- Architectural Design.

  - The role of the software within the system and the relationships between this software and other software components.

  - Assumptions about the environment including operating system, user interface, program, data management, data interchange, graphics, and network services, especially assumptions on which safety functions and computer security needs may be based.

- Architectural Design Description. The design of the software including:

  - Logical or functional decomposition.
  - Description of the modules including mention of the safety and computer security functions.
  - Design of the modules in terms of execution control and data flow including flagging safety and security functions.
  - Design requirements/constraints on the modules.
  - Relationships and interactions between the modules.
  - A schema for the allocation of functions to modules.
  - Logical data design - conceptual schema.
  - Entity/data identification and relationships.
  - Timing and sequencing.
  - Implementation constraints.
  - Each state and mode the software operates in and the modules that execute in each state and mode.
  - Execution control and data flow between modules in the different states and modes.
  - Memory and processing time allocation to the modules.

- External Interface Design. Allocation of the software's external interface requirements to modules. The design of each interface in terms of:

  - Information description.
  - Initiation criteria.
  - Expected response.
  - Protocol and conventions.
  - Error identification, handling and recovery.
  - Queuing.
  - Implementation constraints.
  - Requirements relative to safety and computer security.

- Software requirements allocation to modules.

- Database description. Reference should be made to the document describing the database, its platform, dependencies, etc., or described it here.

- Human interaction. Screens, types of alert mechanisms.

- Tracking to any other systems in the plant that have safety functions.

- Detailed Design.

  - Modules Design and Traceability to Architectural Design. The design of the software into its modules and traceability between the architectural design modules and software modules. The traceability with flags on safety features and computer security requirements. Descriptions of the modules should include:

    - Inputs and outputs.
    - Functions.
    - Data descriptions and relationships.
    - Diagrams.
    - Control and signal flow.
    - Error handling/messages.
    - Interfaces between modules.
    - Packaging details (placement of modules).
    - Flags on safety, computer security functions.

  - Detailed Design of Modules. Design information necessary to code the modules. The information should include:

    - Detailed design to the lowest level.
    - Functions or operations.
    - Algorithms.
    - Specific data definitions including data conversions.
    - Local and global data.
    - Parameters for initiation and adaptation.
    - Logic flow (control flow; timing variations; priority assignments; interrupt priorities and handling).
    - Error detection and handling.
    - Physical data design (internal schema; query language; access method; key, record, and data element definition and structure).
    - Device interface.
    - Interrupts and signals.
    - Limitations that restrict performance of modules.

  - External Interface Detailed Design. The traceability of the external interface design of the software into its modules. A description of each external interface including:

    - Type and purpose.
    - Description of the data transmitted across the interface including purpose, source and destination, data type, data representation, size, units of measure, limit/range, accuracy, precision/resolution.
    - Messages transmitted and the assignment of data elements to each message.

    - Priority of interfaces and messages transmitted across them..
    - Protocol (fragmentation and reassembly of messages; error control and recovery procedures; synchronization; flow control; data transfer rate and minimum transfer rate; routing, addressing and naming conventions; transmission services; status, identification, notification and other reporting features; security).
    - Description of user responses to system/software safety/security violations.

- Coding and Implementation Notes. Information such as stubs for incremental development and use of compiler options.

- Information On Planning, Designing, Executing Tests.

4.6.2. Source Code Documentation

A source code manual should be developed for a project, unless it already exists within the vendor organization, in which case it should be referenced by the SQAP. The source code manual, which may be very general, is used by the programming staff, and possibly also by reviewers when evaluating a product.

SOURCE CODE MANUAL. Description of the standards and practices for coding, including rules on issues such as the format of the code, maintaining the modularity, complexity, data flow of the design, and commenting the code (e.g., explanations of inputs and outputs). Deviations from the rules stated in the manual should be justified in the source code where deviation occurs and in the accompanying documentation.

The purpose of supporting documentation for source code is to describe the source code and enable traceability to requirements, design, and test cases. This documentation is especially useful to the development staff, to new staff members, and to the maintenance staff. Examples of information that might be included in this documentation are the following:

SUPPORTING DOCUMENTATION FOR SOURCE CODE

- Explanation of deviations from the design.

- Explanation of deviations from the source code manual.

- Explanation of deviations from the programming language standard (e.g., the use of a dialect of the language).

- Charts showing the relationship of code modules to design components, or a listing of subroutines that support one design module. This is helpful if the design is at a higher level than the code.

- Interface information, if the design is at higher level than code.
- Identification of the most critical modules for safety and security.
- Comments within modules.
- Description of how "controlled practices" are used (e.g., use of techniques to offset risks of using pointers) [NIST190].

4.7. Test Documentation

Test documentation is a subsection of section 4.4. Software Verification and Validation.

4.8. Installation and Checkout Documentation

During installation and checkout the software system is installed and tested to ensure that it performs as required in its operational environment. Documentation for this phase should include the approval of the software for operational use and may include the user's manual, if installation and checkout is to performed by the end-user, rather than the vendor. See section 4.5.2 for content of the user's manual. The following is a minimum content for an installation and checkout document based on [IEEEP1059].

INSTALLATION AND CHECKOUT DOCUMENT

- Instructions for installation at each site, which accurately and completely represent the actual configuration and are appropriate for the expertise level of the expected installer.
- SV&V activities during installation and checkout (see also section 4.4. V&V).
- Schedule for installation, if software is to be installed at a large number of sites.
- Procedural flowchart or checklist.

4.9. Operation and Maintenance Documentation

O&M documentation should describe procedures (e.g., preparation, monitoring, fault recovery) executed to operate the software, as well as all modifications made to the software during this phase. The following recommendations for the operational procedures manual is based on [NASA2100] and [DOD2167A].

OPERATIONAL PROCEDURES MANUAL

- System Preparation and Set-up Procedures.

    - Power on and off.

    - Initiation. Description of the following:

        - Equipment setup and the procedures required for pre-operation.
        - Procedures necessary to bootstrap the system.
        - Procedures for loading software and data.
        - Procedures for initializing files, variables, or other parameters.
        - Commands typically used during initiation.

- Standard Operating Procedures. Detailed description of standard operating procedures including:

    - Input and output procedures. Description of the following:

        - Input and output media (e.g., magnetic tape, disk, or cartridge). Explain procedures for reading and writing on these media.
        - Operating system control language.
        - Operator procedures for interactive messages and replies (e.g., which terminal to use, passwords, log on/off).

    - Monitoring procedures. Description of procedures for monitoring software during operation, including:

        - Applicable trouble and malfunction indications.
        - Evaluation techniques for fault isolation.
        - Conditions requiring computer system shutdown.
        - On-line intervention, abort, and user communications.

    - Off-line Routine Procedures.

    - Daily operating procedures (e.g., system back-ups, and logs for maintenance).

    - Standard safety and security procedures.

    - On-demand procedures (e.g., in response to user requests).

- Fault and Recovery Procedures. Description of automatic and manual procedures to be followed in case of a fault or abnormal condition in the software. Describe the immediate actions and subsequent recovery procedures for every anticipated trouble condition.

- Emergency Procedures. Description of procedures to be conducted in emergencies, including:

    - Procedures for critical system failures.

    - Environmental emergency procedures (e.g., fires, hurricanes, or thunderstorms).

    - Safety or security emergency procedures.

- Diagnostic Features.

    - Diagnostic Procedures. Identification of each procedure and its purpose. Also description of the following:

        - Hardware, software, or firmware necessary to execute the procedures.
        - Step-by-step instructions for executing the procedure.
        - Diagnostic messages and the corresponding required action.

    - Diagnostic Tools. Description of each tool and its application.

The following recommendations for maintenance documents is based on [EWICS2] and [IEEEP1219].

MAINTENANCE DOCUMENTS. Description of changes made to the software as a result of corrective, adaptive, or perfective maintenance. The modifications should be documented as appropriate in new documents or in addenda to development documents (e.g., SRS, SDD). Various documents are needed to record, track, and implement maintenance. The following are examples of documents that may be used to perform maintenance.

- Anomaly Report. Issued by the operations staff to initiate corrective maintenance.

- Modification Feasibility Request. Request issued by the user management to carry out a feasibility study on a proposed modification. Initiates perfective or adaptive maintenance.

- Feasibility Report. Issued to user management to report on results of the feasibility study.

- Validated Modification Request. Issued by user management either to reject the proposed modification or to proceed.

- Maintenance Action Report. Issued to the user management after completion of the required preventive maintenance detailing the work done.

- Revalidation Request. Request to retest the system after completion of the required preventive maintenance, or implementation of a modification.

- Revalidation and Configuration Management Report. Issued to user management after the necessary revalidation is performed, so user can decide if system should be returned to service.

4.10. Reporting

The purpose of reporting is to communicate to others information on specific deficiencies in the software or to summarize specific tasks or groups of tasks performed during development (e.g., SQA activities and SV&V activities). These reports should be used during formal reviews that assess the current product and development activities, as well as to improve the development process for future projects. Reports should be generated as necessary (e.g., for each anomaly or at the completion of each major task). There is no set number of required reports; however, the vendor should avoid producing an overabundance of unnecessary reports. Examples of necessary reports for an activity during a particular lifecycle phase include task reports and anomaly reports. Examples of similar reports can be found in section 4.4.

5. RECOMMENDATIONS FOR REVIEW AT SAFETY EVALUATION

This section of the report provides recommendations for the NRC review of the safety features of the product. In order to evaluate the product, the reviewers examine all documentation associated with the product and may test the product. Test activities are outside the scope of this study, and are not discussed in this report. Instead, this report focuses on how to evaluate the product based on its documentation. This discussion deals strictly with the evaluation of the software component of the safety system, and not the whole system, although NRC reviewers may examine all components during the safety evaluation. However, the principles identified in this section can be adapted for system reviews, with the addition of system-oriented details.

At the safety evaluation, reviewers are responsible for assessing that the software meets its software quality objectives (e.g., safety, functionality, reliability, maintainability, reviewability) [SOFTENG]. To perform this evaluation, the reviewers need sufficient information about the product requirements, its development, and its overall quality. At a minimum, the reviewers should have available the following: system requirements/description, system hazard analysis report, project planning documentation (e.g., SQAP, SCMP, SVVP, PMP), development documentation (e.g., SRS, SDD, source code), reports of assurance activities, project reports, and formal review reports.

The reviewers will also need to evaluate the installation package, which consists of installation procedures, installation medium (e.g., magnetic tape), test case data used to verify installation, and expected output from the test cases. In some instances, the product may already be installed in the utility. NRC should request documentation on the results of installation and acceptance testing.

O&M documentation should be provided, even though it does not relate directly to the development of the product. The reviewer should check that the operational procedures manual is adequate to ensure that the user will be able to operate the software product safely and correctly. In some instances, the reviewer may be re-evaluating a previously-approved product, which has undergone extensive maintenance. For these reviews, the vendor should provide the maintenance documentation, which describes all modifications made to the original product during the O&M phase.

An assumption about the NRC review process is that the reviewers may examine some documentation prior to the site visit, where the bulk of the review occurs. The site may be either that of a vendor or utility, depending on the project. If the review occurs at the vendor's site, the reviewer may request interviews with vendor personnel as needed. The review process involves primarily the review of documentation, but not at the same level of detail as the formal reviews conducted during the development process. NRC reviewers focus on how the product was built, on the results of assurance activities, and the tracking of a selected set of inputs through the system (a string check), rather than a detailed examination of the product. NRC reviewers, while interested in the answers to the checklist questions used in formal reviews, are more concerned with how the vendor responded to the findings of the formal reviews. NRC should request results of all vendor/utility assurance activities, and responses to problems. The NRC reviewers may use sections 4 and Appendix B of this report as baselines for examining the documentation. While the reviewers may perform some dynamic or static analyses on the code, discussion of these activities is outside the scope of this task.

The overall purpose of the safety evaluation review (from the software perspective) is to assess how well the product meets its software quality objectives. Some general types of questions regarding the product, such as the following, guide the reviewers:

- How thoroughly has the vendor analyzed the safety of critical functions of the software?
- How well has the vendor established the appropriateness of the functions, algorithms, and knowledge on which the software is based?
- How carefully has the vendor implemented the safety and performance requirements of the software?
- If any function of the safety system was performed by previous systems (either hardware, firmware, or software), is the software substantially equivalent to the predicate device?

The types of questions listed in the remainder of this section provide guidelines for the safety evaluation and are not a complete list. At any point, the reviewer may find it necessary to explore a topic in more detail, and may refer to the checklists in Appendix B. The checklists for this section are based on [HHSFDA], [IEC880], [EWICS2], [ANS104], and [PARNAS].

<u>Requirements and Design</u>

- Does the SRS identify safe and unsafe operating states?
- Does the SRS identify potentially unsafe operational procedures?
- Does the SRS identify measures for monitoring potentially unsafe situations?
- What are the safety features and the rationale for them?
- Are the safety requirements consistent with the hazard analysis?
- Do the safety requirements address both environmental influences and random component failures?
- Have the different classes of users been considered?
- Are the enforced constraints (legal, company-related) included in the safety criteria?
- Does the SRS describe constraints between hardware and software?
- Were the functional requirements analyzed for intrinsic hazards?
- Are functional requirements classified according to safety criticality?
- Does the SRS identify requirements for redundant controls between electro-mechanical and software controls?
- Are there requirements for self supervision of the software? Do they meet safety requirements?
- Is there modularity in the design?
- Have critical components been isolated?

Software Development Methodologies

- Were formal methods used? If so, how much experience does the vendor have with formal methods?
- Were design reviews performed and documented?
- How was failure analysis of the software conducted? What methods were used (e.g., FMECA)?
- If automated tools were used, were they thoroughly tested for adequacy before use?
- Was a high level language with a tested translator used?

Installation Package

- Does the package contain sufficient materials to permit rebuilding and testing of the installed program?
- Are the installation procedures clearly understandable? Are the format and content of the tape properly identified?
- Can the program be rebuilt from the installation package?
- Do the test cases produce results identical with the expected outputs?

Operational Procedures Manual

- Does the manual adequately describe standard operating procedures, as well as emergency procedures, and fault and recovery procedures?
- Does the manual adequately describe diagnostic procedures and tools?

Maintenance[6]

- Are all modifications made to the software during maintenance documented?
- Is there a formal procedure to start maintenance when problems are found?
- Is there a standard procedure to treat all modification requests?
- Is there a common modification request document for all disciplines?
- Is there a formal change control procedure?
- Are modification tools (e.g., diagnostic tools, configuration management tools) addressed, if any are used?

Assurance Activities

Project Management:
- Were reviews of project status and assurance task reports conducted?
- Based on metrics data, were any processes found to be outside statistical control limits? If so, what actions were taken?

[6] This section may not be applicable for all safety evaluations. However, if maintenance documentation is provided, the reviewer should consider these questions.

SV&V:

- What analysis techniques were used (e.g., control flow analysis, inspection)?
- Were the test strategies consistent with the requirements and were they adequate to evaluate the design?
- Were the test strategies sufficient to determine whether the software is safe and effective?
- How were the safety features of the software tested and analyzed?
- Do the results of testing and analysis demonstrate conformance to requirements established at each level (i.e., module, integration, system)?
- Do test results show that safety related functions were tested adequately and that they perform reliably?

SCM:

- What SCM facilities were used?
- At what point was the first SCM baseline established?
- What evidence is there that the software delivered is the software that has been verified and validated?

SQA:

- What metrics were used, if any?
- What actions were taken in response to the metrics results?
- Do SQA reports indicate that standards were appropriately followed?
- What error analysis techniques were used? How did vendor respond to the findings?
- Were software design and code inspections performed? If so, were the identified problems corrected?
- Were technical reviews held, and problems identified in the reviews satisfactorily resolved in the product?

6. SUMMARY

[ASMENQA2] addresses activities for software lifecycle phases from software requirements to software maintenance, as well as some assurance activities, but does not necessarily require documentation or review of all these activities. Some SCM documentation is identified, but there is no requirement for an SCMP. Requirements for quality management, project management, and SQA elements including measurement, risk management, SQA audits and training are also not addressed. Although [ASMENQA2] does require documentation of many lifecycle and assurance activities (e.g., SVVP, SQAP, SCM documentation), the requirements are either very general or incomplete.

Because [ASMENQA2] is based on IEEE's first SQA standard, IEEE Std 730-1984, which was written six years before [ASMENQA2] and has since been revised, it fails to address current issues, such as software safety and computer security. The omission of requirements for project management, an SCMP, and metrics, as stated above may also be attributed to this fact. Since [ASMENQA2] does not specify unique assurance requirements for safety and security, this report identifies documentation requirements that emphasize these issues, as well as other important issues that are not addressed by [ASMENQA2]. This report also identifies review issues concerning software properties that may affect the safety of the total system.

One feature of [ASMENQA2] that may be confusing is its order of presentation. First, it describes the activities of each lifecycle phase. It then lists the documentation for each phase. Finally, the reviews of each lifecycle phase are addressed. Although no preference for any particular format is specified, one which may be more logical is to provide only one section for each lifecycle phase or assurance activity, and to specify the requirements for the activity, its documentation, and its review all in the same section.

Another feature of [ASMENQA2] that may be of concern is its lack of requirements specific to nuclear applications. Although its title indicates that it is an SQA standard intended for nuclear facility applications, [ASMENQA2] does not seem to be tailored for this purpose. Rather, it resembles a general SQA standard that could be applied to any software system.

For each software life cycle activity, this report provides recommended content for documentation of the activity. The activities include not only the development phases from software requirements to software maintenance but also project management, SCM, SV&V (including testing), and SQA.

While this report concentrates on activities for the *software* lifecycle, it also identifies some issues that are of special interest when the software lifecycle is embedded within the *system* lifecycle. One issue concerns information from the system requirements phase that are essential for the software development phases. Another issue concerns the scope and definition of *software* verification and validation within system verification and validation.

This report distinguishes between the formal reviews conducted during development and those conducted by NRC. The generic procedures for formal development reviews are provided in Appendix B, with detailed lists of questions for specific reviews. For these reviews, it is assumed that their purpose is to enact changes to the development process and product under review. For reviews conducted by NRC, it is assumed that these concentrate more on how the product was built, on the results of assurance activities, and the tracking of a selected set of inputs through the system (a string check), rather than a detailed examination of the product. Hence, questions for NRC reviewers are at a much higher level of abstraction than for development reviews, but the NRC reviewers may request the results of the vendor's formal reviews and may also use the detailed checklists to support their reviews as appropriate.

Although the required scope for this study includes only safety systems in nuclear power plants, the recommendations for documentation and reviews are presented in a broader context to ensure completeness. While the recommendations may be applied to systems of varying size, the purpose in expanding the scope is to provide NRC reviewers with a comprehensive understanding of what comprises a quality project. The important issue is not the exact number of documents, but rather the *content* of the different types of documentation. For small projects, the documents may be combined, but the necessary content should be retained. Thus, depending on the product under review, NRC may decide that not all the recommendations are appropriate and may add additional safety-related review items to improve the coverage of safety issues.

7. REFERENCES

[ANS104]
ANSI/ANS-10.4-1987, "Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry," American Nuclear Society, May 13, 1987.

[APPGUIDE]
Application Portability Profile (APP) The U.S. Government's open System Environment Profile OSE/1 Version 1.0, The National Institute of Standards and Technology, NIST SP 500-187, 1991.

[ASMENQA2]
ASME NQA-2a-1990 Part 2.7, "Quality Assurance Requirements for Nuclear Facility Applications," The American Society of Mechanical Engineers, November 1990.

[BIRRELL]
Birrell, N. and Ould, M., A Practical Handbook For Software Development, Cambridge University Press, 1985.

[DOD2167A]
DOD-STD-2167A, "Defense System Software Development," United States Department of Defense, 29 February 1988.

[DUNN]
Dunn, Robert, Software Quality: Concepts and Plans, Prentice Hall, 1990.

[EWICS2]
Redmill, F. J. (ed.), Dependability of Critical Computer Systems 2, The European Workshop on Industrial Computer Systems Technical Committee 7 (EWICS TC7), Elsevier Science Publishers LTD, 1989.

[FUJII]
Fujii, Roger U., Dolores R. Wallace, Taz Daughtrey, "Software Verification and Validation Seminar," IEEE Standards Seminar, 1990.

[HHSFDA]
"Reviewer Guidance for Computer Controlled Medical Devices Undergoing 510(k) Review," Office of Device Evaluation, Center for Devices and Radiological Health, Food and Drug Administration.

[IEC880]
IEC 880, "Software for Computers in the Safety Systems of Nuclear Power Stations," International Electrotechnical Commission, 1986.

[IEEEP1059]
IEEE Std P1059, "(DRAFT 6.3) IEEE Guide for Software Verification and Validation Plans," The Institute of Electrical and Electronics Engineers, Inc., December 26, 1991.

[IEEEP1219]
IEEE Std P1059, "Standard for Software Maintenance," The Institute of Electrical and Electronics Engineers, Inc., January 31, 1992.

[IEEE610]
ANSI/IEEE Std 610.12-1990, "Glossary of Software Engineering Terminology," The Institute of Electrical and Electronics Engineers, Inc., 1990.

[IEEE828]
ANSI/IEEE Std 828-1983, "IEEE Standard for Software Configuration Management Plans," The Institute of Electrical and Electronics Engineers, Inc., 1984.

[IEEE830]
ANSI/IEEE Std 830-1984, "IEEE Guide for Software Requirements Specifications," The Institute of Electrical and Electronics Engineers, Inc., 1984.

[IEEE1012]
ANSI/IEEE Std 1012-1986, "IEEE Standard for Software Verification and Validation Plans," The Institute of Electrical and Electronics Engineers, Inc., November 14, 1986.

[IEEE1028]
ANSI/IEEE Std 1028-1988, "IEEE Standard for Software Reviews and Audits," The Institute of Electrical and Electronics Engineers, Inc., 1989.

[IEEE1058]
ANSI/IEEE Std 1058.1-1987, "IEEE Standard for Software Project Management Plans," The Institute of Electrical and Electronics Engineers, Inc., 1988.

[IEEE7300]
ANSI/IEEE Std 730-1984, "IEEE Standard for Software Quality Assurance Plans," The Institute of Electrical and Electronics Engineers, Inc., 1984.

[IEEE7301]
ANSI/IEEE Std 730.1-1989, "IEEE Standard for Software Quality Assurance Plans," The Institute of Electrical and Electronics Engineers, Inc., October 10, 1989.

[ISOSC7]
ISO/IEC(JTC1)-SC7, "Information Technology Software Life-Cycle Process," August 30, 1991.]

[ISO9000]
ISO 9000, "International Standards for Quality Management," International Standards Organization, ISO Central Secretariat, Casepostale 56, CH-1211, Geneve 20, Switzerland, May 1990.

[NASA2100]
NASA-STD-2100-91, "NASA Software Documentation Standard - Software Engineering Program," National Aeronautics and Space Administration, July 29, 1991.

[NIST190]
NIST Special Publication 500-190, "Proceedings of the Workshop on High Integrity Software; Gaithersburg, MD; Jan. 22-23, 1991," U.S. Department of Commerce/National Institute of Standards and Technology, August 1991.

[NIST204]
NIST Special Publication 500-204, "High Integrity Software Standards and Guidelines," U.S. Department of Commerce/National Institute of Standards and Technology, September 1992.

[NUREG5930]
NUREG/CR-5930, "High Integrity Software Standards and Guidelines," U.S. Nuclear Regulatory Commission, September 1992.

[PARNAS]
Parnas, D., "Active Design Reviews: Principles and Practice," IEEE International Conference on Software Engineering, London, 1985, p. 132-136.

[SEI91]
Paulk, Mark C., Bill Curtis, and Mary Beth Chrissis, "Capability Maturity Model for Software," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, CMU/SEI-91-TR-24, ESD-TR-91-24, August, 1991.

[SOFTENG]
"Standard for Software Engineering of Safety Critical Software," Draft, Rev. 0, Ontario Hydro, December 1990.

[STARTS]
STARTS Purchasers Group, "The STARTS Purchasers' Handbook," NCC Publications, October 1987.

APPENDIX A. DEFINITIONS OF QUALITY ATTRIBUTES

The following definitions include direct quotes from [IEEE610] and [SOFTENG].

compatibility	(1) The ability of two or more systems or components to perform their required functions while sharing the same hardware or software environment. (2) The ability of two or more systems or components to exchange information.
completeness	The degree to which all of the software's required functions and design constraints are present and fully developed in the SRS, SDD, and code.
consistency	The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component. See also TRACEABILITY.
correctness	(1) The degree to which a system or component is free from faults in its specification, design, and implementation. (2) The degree to which software, documentation, or other items meet specified requirements. (3) The degree to which software, documentation, or other items meet user needs and expectations, whether specified or not. (4) The ability of the SRS, SDD, and code to describe or produce the specified outputs when given the specified inputs, and the extent to which they match or satisfy the requirements.
feasibility	The degree to which the requirements, design, or plans for a system or component can be implemented under existing constraints.
modifiability	Quality attribute that refers to the characteristics of the SRS, SDD, and code which facilitate the incorporation of changes.
modularity	The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
predictability	The degree to which the functionality and performance of the software are deterministic for a specified set of inputs.
robustness	The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

structuredness	The degree to which the SDD and code possess a definite pattern in their interdependent parts. This implies that the design has proceeded in an orderly and systematic manner (e.g., top-down), has minimized coupling between modules, and that standards control structures have been followed during coding resulting in well structured software.
testability	(1) The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. (2) The degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met.
traceability	(1) The degree to which a relationship can be established between two or more products of the development process. (2) The degree to which each element in a software development product establishes its reason for existing (e.g., the degree to which each element in a bubble chart references the requirement that it satisfies). (3) The ability to trace the design decision history and reasons for these changes. See also CONSISTENCY.
understandability	The degree to which the meaning of the SRS, SDD, and code are clear to the reader.
verifiability	The degree to which the SRS, SDD, and code have been written to facilitate verification using both static methods and testing.

APPENDIX B. FORMAL REVIEWS OF THE DEVELOPER

The purpose of these reviews is to enact changes to the development process and product under review. For reviews conducted by NRC, the purpose is to concentrate more on how the product was built, on the results of assurance activities, and on tracking of behavior of a selected string, or slice, through the product rather than a detailed examination of the product. Hence, questions for NRC reviewers are at a much higher level of abstraction than for development reviews, but the NRC reviewers may use the detailed checklists.

Some standards that address SQA assume a contractual environment or an environment that can be treated contractually [IEEE7301, ISOSC7]. One of the requirements that generally appears in such standards is that both the developer and the customer must attend/conduct the formal reviews. An environment that is treated contractually is one in which the developer's organization may have several principal organizations within it (e.g., marketing, design, systems engineering, programming, SQA) and those persons developing the software may be responsible for formal reviews with other members of the organization. Typical formal reviews that are conducted between developer and customer or among organizations of the developer are discussed in this section of the report. NRC reviewers should be familiar with these formal review procedures, their results, and the follow-up results of specific product reviews. NRC should recommend that the utilities provide (or have available for) NRC the results of formal reviews and of follow-up activities. This is especially important because NRC reviewers at safety evaluation may want to check whether or not formal reviews were performed (see section 5), and may use review reports to examine how well the product was built.

Formal reviews are conducted at the end of each lifecycle phase or at the end of the planning period on the results or products of the phase or period. They may also be conducted when a serious problem or concern arises. This appendix to the report summarizes the key factors of management and technical reviews (document evaluation), discusses the role assurance activities take in the formal review process, and provides sample checklists for the technical/development reviews. For completeness all the technical/development reviews listed in Table 3-1 are included, even though [ASMENQA2] does not address them all.

B.1. The Review Process

Management reviews formally evaluate a project plan or project status relative to that plan. Management reviews have two purposes. The first is to ensure the adequacy and completeness of each planning document (e.g., PMP, SQAP, SCMP, SVVP, and Test Plans) for meeting project requirements. The second is to ensure that project activities are progressing according to the planning documents, identify the need for corrective action to the plan or the project, and ensure proper allocation of resources. All problems are documented. The results of the management reviews are summarized in a management review report which are auditable and traceable to and from the appropriate planning documents.

In contrast, the formal technical review examines the product, and the results of any assurance activities already conducted on the product. The purpose of technical reviews is to evaluate the software elements (e.g., SRS, software design description (SDD)) to ensure conformity to its specifications, compliance of the development of the software elements with its plans, and the integrity of changes to the software elements. The results of the technical reviews are summarized in technical review reports which are auditable and traceable to and from the appropriate planning documents. Success of a technical review requires that all participants carefully examine the inputs to the technical review prior to the review meeting. Section B.1.1 presents an outline of the management review process; section B.1.2 presents an outline of the technical review process. The descriptions of management and technical reviews are based on [IEEE1028].

In both the management and technical reviews, experts on specific topics (e.g., design experts for design reviews) should be present to lend their expertise to the review.

B.1.1. Management Reviews

<u>Responsibilities</u>. The review leader performs the administrative functions of the review and issues the management review report. Review team members are expected to be prepared for the meeting and ensure that the review objectives are met.

<u>Inputs</u>. Objectives of the review; list of issues to discuss; the specific planning document; current project schedule and cost data; reports from previously completed reviews; reports on project resources; and, data on complete or in progress software elements.

<u>Entry Criteria</u>. The need for conducting management reviews is specified in the appropriate project planning documents. Other management reviews may also be conducted upon request. The management review is performed when the review leader establishes/confirms the review objectives and determines that all necessary documents are available.

<u>Procedures</u>. The review leader and project manager plan for the review by identifying the review team, scheduling a time and place for the review, and distributing all inputs to the review team. An overview of the project is conducted for the review team by a qualified project member. Each review team member studies the inputs and prepares presentations for the review team. The management review consists of the review team:

- Assessing adequacy and completeness of the planning documents (initial review).
- Determining whether or not the project status corresponds to the planning document under review, and recording any deviations.
- Determining whether or not factors not originally considered are constraining the planning document under review.
- Listing issues and recommendations to be addressed by upper management and/or others who affect the project.

- Recommending actions to follow the review, and authorization for additional reviews and audits.
- Identifying other issues that need to be addressed.

Exit Criteria. The management review is complete once the review objectives have been addressed and the management review report has been issued.

Output. The management review report identifies: the project; the review team; review inputs; review objectives; action items; list of issues and recommendations; and, recommendations for additional reviews and information necessary to complete them. (The procedures for closing action items should be part of a plan.)

B.1.2. Technical Reviews

Responsibilities. The review leader performs the administrative functions of the review and issues the technical review report. The recorder documents the findings, decisions, and recommendations of the review team. Review team members are expected to be prepared for the meeting and ensure that the review objectives are met. Recommendations made by the review team should be such that management can act on them quickly. Management is responsible for responding to recommendations promptly.

Inputs. Objectives of the review; software element being reviewed; the software element's specifications; results of any assurance activities; any other necessary documentation.

Entry Criteria. The need for conducting technical reviews is specified in the project planning documents. Other technical reviews may also be conducted upon request. The technical review is performed when the review objectives are established, the individuals responsible at the review for the software element are prepared for the review, and the review leader determines the software element is sufficiently complete. Preparation may require considerable time spent in privately examining the inputs.

Procedures. The review leader plans for the review by identifying the review team, schedules a time and place for the review, and distributes all inputs to the review team. An overview of the project is conducted for the review team by a qualified project member. Each review team member studies the software element and related materials. The technical review consists of the review team:

- Determining whether or not the software element corresponds to the specifications and standards to which it must adhere, and recording any deviations.
- Listing issues, recommendations, and responsible individuals for resolving the issues.

- Identifying other issues that need to be addressed.
- Documenting the meeting, deficiencies found it the software element, and recommendations for management (the review leader determines whether or not an additional review should be performed on reworked software elements).

Exit Criteria. The technical review is complete once the review objectives have been addressed and the technical review report has been issued.

Output. The technical review report identifies: the review team; the software element; review inputs; the software element's unresolved deficiencies; list of management issues; action items; and, recommendations for unresolved issues deficiencies.

V&V activities should be performed prior to the formal review of the product of each of the lifecycle phases. The reviewer during development checks that these activities have been performed, and uses the resulting reports to answer the questions in the checklists below.

B.2. Checklists for Formal Reviews

Formal reviews may include reviewing SQA, SV&V, and SCM results in order to examine the product. This review may also help detect whether or not these activities were performed in accordance with their respective plans. The checklists below provide a guideline for reviewing both the product and plans for assurance activities. These checklists are not necessarily complete; at any time the reviewer during development may need to expand upon a given topic. In all checklists, negative answers require further examination by the reviewers.[7]

B.2.1. Software Requirements Review

The following checklist contains questions a reviewer during development ask at the software requirements review based on [IEEE1028], [STARTS], [SOFTENG], [EWICS2], [IEEEP1059], [HHSFDA], [BIRRELL], and [ANS104].

Compatibility[8]

- Do the interface requirements enable compatibility of external interfaces (hardware and software)?

[7]Some details of this further examination will be provided in Task 4, Error Analysis.

[8]Definitions for quality attributes in the checklists appear in Appendix A.

Completeness

- Does the SRS contain everything listed in the corresponding documentation content? Does it include all requirements relating to functionality, performance, constraints, safety, etc.?
- Does SRS include all user requirements (as defined in the concept phase)?
- Do the functional requirements cover all abnormal situations?
- Have the temporal aspects of all functions been considered?
- Are the time-critical functions identified and the time criteria for them specified? Do these include the maximum and minimum times for their execution?
- Does SRS define those requirements for which future changes are anticipated?
- Are all normal environmental variables included?
- Are the environmental conditions specified for all operating modes (e.g., normal, abnormal, disturbed)?

Consistency

- Is there internal consistency between the software requirements?
- Is the SRS free of contradictions?
- Are the specified models, algorithms, and numerical techniques compatible?
- Does SRS use standard terminology and definitions throughout?
- Is SRS compatible with the operational environment of the hardware and software?
- Has the impact of software on the system and environment been specified?
- Has the impact of the environment on the software been specified?

Correctness

- Does the SRS conform to SRS standards?
- Are algorithms and regulations supported by scientific or other appropriate literature?
- What evidence is there that shows vendor has applied the regulations correctly?
- Does the SRS define the required responses to all expected types of errors and failure modes identified by the hazard analysis?
- Were the functional requirements analyzed to check if all abnormal situations are covered by system functions?
- Does SRS reference desired development standards?
- Does the SRS identify external interfaces in terms of input and output mathematical variables?
- Are the requirements for the man-machine interface adequate?
- What is the rationale for each requirement? Is it adequate?
- Is there justification for the design/implementation constraints?

Feasibility

- Will the design, operation, and maintenance of software be feasible?
- Are the specified models, numerical techniques, and algorithms appropriate for the problem to be solved? Are they accepted practice for nuclear power plants? Can they be implemented within the imposed constraints?
- Are the quality attributes specified achievable individually and as a group?

Modifiability

- Are requirements organized so as to allow for modifications (e.g., with adequate structure and cross referencing)?
- Is each unique requirement defined more than once? Are there any redundant statements?
- Is there a set of rules for maintaining the SRS for the rest of the software lifecycle?

Robustness

- Are there requirements for fault tolerance and graceful degradation?

Traceability

- Is there traceability from the next higher level spec (e.g., system concept/requirements and user needs as defined in concept phase, and system design)?
- Does the SRS show explicitly the mapping and complete coverage of all relevant requirements and design constraints defined in the concept phase, by such means as a coverage matrix or cross-reference?
- Is SRS traceable forward through successive development phases (e.g., into the design, code, and test documentation)?
- Are safety functions or computer security functions flagged?

Understandability

- Does every requirement have only one interpretation?
- Are the functional requirements in modular form with each function explicitly identified?
- Is there a glossary of terms?
- Is formal or semiformal language used?
- Is the language ambiguous?
- Does the SRS contain only necessary implementation details and no unnecessary details? Is it over specified?
- Are the requirements clear and specific enough to be the basis for detailed design specs and functional test cases?
- Does the SRS differentiate between program requirements and other information provided?

Verifiability/Testability

- Are the requirements verifiable (i.e., can the software be checked to see whether requirements have been fulfilled)?
- Are mathematical functions defined using notation with precisely defined syntax and semantics?
- Is there a verification procedure defined for each requirement in the SRS?

B.2.2. Software Design Review

Reviewers should be able to determine whether or not all design features are consistent with the requirements. Numerical techniques and algorithms should be appropriate for the problem to be solved. The program design needs to be partitioned in a manner consistent with the problem to be solved. And, the program should meet the requirements. The following checklist contains questions a reviewer during development may ask at the software design review based on [SOFTENG], [IEEE1028], [IEEEP1059], and [ANS104].

Completeness

- Are all the items listed in section 4.6.1. addressed in the SDD?
- Are the SRS requirements fulfilled?
- Is there enough data (logic diagrams, algorithms, storage allocation charts, etc.) available to ensure design integrity?
- Are algorithms and equations adequate, accurate, and complete?
- Are requirements for the support and test software and hardware to be used in the development of the product included?
- Does the design implement required program behavior with respect to each program interface?
- Are all program inputs, outputs, and database elements identified and described to the extent needed to code the program?
- Does the SDD describe the operational environment into which the program must fit?
- Are all required processing steps included?
- Are all possible outcomes of each decision point designated?
- Does the design take into account all expected situations and conditions?
- Does the design specify appropriate behavior in the face of unexpected or improper inputs and other anomalous conditions?
- Does the SDD reference all desired programming standards?

Consistency

- Are standard terminology and definitions used throughout the SDD? Are the style of presentation and the level of detail consistent throughout the document.
- Does the design configuration ensure integrity of changes?
- Is there compatibility of the interfaces?

- Is the test documentation compatible with the test requirements of the SRS?
- Is the SDD free of internal contradictions?
- Are the models, algorithms, and numerical techniques that are specified mathematically compatible?
- Are input and output formats consistent to the extent possible?
- Are the designs for similar or related functions consistent?
- Are the accuracies and units of inputs, database elements, and outputs that are used together in computations or logical decisions compatible?

<u>Correctness</u>

- Does the SDD conform to design documentation standards?
- Does the design perform only that which is specified in the SRS unless additional functionality is justified?
- Is the test documentation current and technically accurate?
- Is the design logic sound -- will the program do what is intended?
- Is the design consistent with documented descriptions and know properties of the operational environment into which the program must fit?
- Do interface designs agree with documented descriptions and known properties of the interfacing elements?
- Does the design correctly accommodate all inputs, outputs, and database elements whose format, content, data rate, etc. are not at the discretion of the designer?

<u>Feasibility</u>

- Are the specified models, algorithms, and numerical techniques accepted practices for use within nuclear power plants?
- Can they be implemented within the constraints imposed on the system and on the development effort?
- Are the functions as designed implementable within the available resources?

<u>Modifiability</u>

- Does the design use information hiding as follows:
  - ▸ The modules are organized such that changes in the requirements only require changes to a small number of modules.
  - ▸ Functions and data structures likely to changes have interfaces insensitive to changes in individual functions.
  - ▸ The design partitions data structure access, database access and I/O access from the application software by the use of access programs (globally accessible data is not used).
  - ▸ Functionality is partitioned into programs to maximize the internal cohesion of programs and to minimize program coupling.
- Does each program have a single function?

Modularity

- Is there a schema for modularity, e.g., model-based?
- Is the design structured so that it comprises relatively small, hierarchically related programs or sets of programs, each performing a particular, unique function?
- Does the design use specific criteria to limit program size?

Predictability

- Does the design contain programs which provide the required response to identified error conditions?
- Does the design schedule computer resources in a manner that is primarily deterministic and predictable rather than dynamic?
- Does the design contain a minimum number of interrupts and event driven software? Is justification given for uses of these features?
- Is plausibility checking performed on the execution of programs to uncover errors associated with the frequency and/or order or program execution and the permissiveness of program execution?

Robustness

- Are all SRS requirements related to fault tolerance and graceful degradation addressed in the design?

Structuredness

- Does the design use a logical hierarchical control structure?

Traceability

- Does the SDD show mapping and complete coverage of all requirements and design constraints in the SRS?
- Are all functions in the SDD outside the scope of the SRS identified?
- Are all functions identified so they can be uniquely reference by the code?
- Does the SDD contain or reference a revision history which identifies all modifications to the design and the rationale for these changes?
- Does the SDD reference the design notes which document design decisions relevant to the software design?
- Have safety and computer security functions been flagged?

Understandability

- Does the SDD avoid unnecessarily complex designs and design representations.
- Is the SDD written to allow unambiguous interpretation?

Verifiability/Testability

- Does the SDD describe each function using well-defined notation so that the SDD can be verified against the SRS and the code can be verified against the SDD?
- Are conditions, constraints identified quantitatively so that tests may be designed?

B.2.3. Source Code Review

The following checklist contains the kinds of questions a reviewer during development may ask at the source code review based on [SOFTENG], [ANS104], and [EWICS2].

Completeness

- Is the code a complete and precise implementation of the design as documented in the SDD?
- Was the code integrated and debugged to satisfy the design specified in the SDD?
- Does the code create the required databases, including the appropriate initial data?
- Are there any unreferenced or undefined variables, constants, or data types?

Consistency

- Is the code logically consistent with the SDD?
- Are the same format, invocation convention, and structure used throughout?

Correctness

- Does the code conform to specified standards?
- Are all variables properly specified and used?
- Are all comments accurate?
- Are all programs invoked with the correct number of parameters?

Modifiability

- Does the code refer to constants symbolically to facilitate change?
- Are cross-references or data dictionaries included to show variable and constant access by the program?
- Does code consist of programs with only one entry point and one exit point? (exception is with fatal error handling)
- Does code reference labels or other symbolic constants rather than addresses?

Predictability

- Is the code written in a language with well-defined syntax and semantics:
- Was the use of self-modifying code avoided?

- Does the code avoid relying on defaults provided by the programming language?
- Is the code free of unintended infinite loops?
- Does the code avoid recursion?

Robustness

- Does the code protect against detectable runtime errors (e.g., range array index values, division by zero, out of range variable values, and stack overflow)?

Structuredness

- Is each function of the program recognizable as a block of code?
- Do loops only have one entrance?

Traceability

- Does the code identify each program uniquely?
- Is there a cross-reference framework through which the code can be easily and directly traced to the SDD?
- Does the code contain or reference a revision history of all code modifications and the reason for them?
- Have all safety and computer security functions been flagged?

Understandability

- Do the comment statements adequately describe each routine, using clear English language?
- Were ambiguous or unnecessarily complex coding used? If so, are they clearly commented?
- Were consistent formatting techniques (e.g., indentation, use of white space) used to enhance clarity?
- Was a mnemonic naming convention used? Does the naming reflect the type of variable?
- Is the valid range of each variable defined?
- Does the code use mathematical equations which correspond to the mathematical models described/derived in the SDD?

Verifiability

- Are implementation practices and techniques that are difficult to test avoided?

B.2.4. Test Readiness Review

The test readiness review is usually conducted following completion of component testing or software integration testing. The purpose is to ensure readiness to begin formal integration testing or system testing without complications, and to ensure that test documentation is complete, that errors have been removed, and that use of the test facilities has been planned properly. The following are general questions that might be asked at these reviews:

- Have recommended changes been made to the code as result of source code review or, as appropriate, component test or integration test?
- Is the error rate sufficiently low to warrant beginning the next type of testing?
- Are all test cases/procedures complete?
- Is the test facility ready? Are schedules approved, personnel and physical requirements specified?
- Have all test tools been checked?
- Have all test procedures been checked?

B.2.5. Test Report Review

The purpose of this review is to assure the completeness of test activities. It is usually required at the end of development (i.e., following system testing). However, it may be conducted after completion of module, integration, or system testing.

The reviewer should ask questions about whether the test objectives were met (e.g., whether all test cases were executed, whether test results matched the expected outputs, and whether the results were traceable to the test cases and software requirements). The reviewer should also inquire about the nature of major anomalies (e.g., whether there were similarities, what corrective actions were taken). The reviewer should check that the anomalies existed only in the product, and were not caused by the test cases and procedures.

B.2.6. Development Documentation Review

[ASMENQA2] states that a development documentation review should be conducted following completion of the testing phase, to "assure completion and acceptability of the development documentation." Although [ASMENQA2] does not define development documentation, it is assumed that this consists of all documentation produced during the development phases of the lifecycle (e.g., requirements, design, coding). The purpose of the development documentation review seems to be checking for consistency among the development documents and to ensure that all changes have been made consistently.

B.2.7. Installation and Checkout Review

The following checklist contains the kinds of questions a reviewer during development may ask at the installation and checkout review based on [ANS104].

Completeness

- Are elements necessary for rebuilding and testing of the installed program available on the installation medium (e.g., source code, user-supplied library routines, test cases)?
- Has the vendor provided adequate information for installing the program in more than one operating environment?

Correctness

- Can all test cases be performed?
- Do the test cases produce results identical to the expected outputs?
- Are all results identical to previous results (when the same tests are executed more than once)? If not, are differences in results clearly understood and justified?

Understandability

- Are the format and content of the medium properly identified for easy reading of the files?
- Are the installation procedures clearly understandable?

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4909

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
SEPTEMBER 1992

4. TITLE AND SUBTITLE

Software Quality Assurance: Documentation and Reviews

5. AUTHOR(S)

Dolores Wallace, Wendy Peng, Laura Ippolito

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

This study examines the contents of a software quality assurance standard for nuclear applications. The study includes recommendations for the documentation of software systems. Background information on the standard, documentation, and the review process is provided. The report includes an analysis of the applicability, content, and omissions of the standard and compares it with a general software quality assurance standard produced by the Institute for Electrical and Electronics Engineers. Information is provided for the content of the different types of documentation. This report describes information for use in safety evaluation reviews. Many recommendations in this report are applicable for software quality assurance in general.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

Documentation; Nuclear Power Plants; Project Management; Reviews, Safety Evaluation; Software Configuration Management; Software Design; Software Quality Assurance; Software Requirements; Software Safety; Software Verification and Validation; Test

13. AVAILABILITY

[X] UNLIMITED
[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
65

15. PRICE
A04

ELECTRONIC FORM