NEW

Branch Technical Position HICB-14

Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems

A. Background

The Staff's acceptance of software for safety system functions is based upon (1) confirmation that acceptable plans were prepared to control software development activities, (2) evidence that the plans were followed in an acceptable software life cycle, and (3) evidence that the process produced acceptable design outputs. This branch technical position (BTP) provides guidelines for evaluating software life-cycle processes for digital computer-based instrumentation and control (I&C) systems. These guidelines are based on reviews of licensee submittals, EPRI's requirements for advanced reactor designs, and the analysis of standards and practices documented in NUREG/CR-6101, "Software Reliability and Safety in Nuclear Reactor Protection Systems." The structure of this BTP is derived from the review process described in Appendix 7.0-A.

1. Regulatory Basis

10 CFR 50.55a(h) requires in part that protection systems satisfy the criteria of ANSI/IEEE Std 279, "IEEE Standard Criteria for Protection Systems for Nuclear Power Generating Stations." Paragraph 4.3 of ANSI/IEEE Std 279 states in part that quality of components is to be achieved through the specification of requirements known to promote high quality, such as requirements for design, inspection, and test. Similar criteria for the quality of components are identified in IEEE Std 603, "IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations."

10 CFR 50, Appendix A, General Design Criterion (GDC) 1, "Quality Standards and Records," requires in part that systems and components important to safety be designed, fabricated, erected, and tested to quality standards commensurate with the importance of the safety functions to be performed. Where generally recognized codes and standards are used, they should be identified and evaluated to determine their applicability, adequacy, and sufficiency, and should be supplemented or modified as necessary to ensure a quality product consistent with the required safety function.

10 CFR 50 Appendix A, GDC 21, "Protection System Reliability and Testability," requires in part that protection systems be designed for high functional reliability commensurate with the safety function to be performed.

10 CFR 50, Appendix B, Criterion III, "Design Control," requires in part that quality standards be specified and that design control measures be provided for verifying or checking the adequacy of design. Criterion V, "Instructions, Procedures, and Drawings," requires in part that activities affecting quality should be prescribed by "documented. . . procedures. . . of a type appropriate to the circumstances. . . ." This BTP outlines such procedures for software. Further, Criterion VI, "Document Control," requires in part that "measures should be established to control the issuance of documents. . . which prescribe all activities affecting quality. . . . These measures should ensure that documents, including changes, are reviewed for adequacy and approved for release by authorized personnel. . . ."

2. Relevant Guidance

Reg. Guide 1.152, "Criteria for Digital Computers in Safety Systems of Nuclear Power Plants," which endorses IEEE Std 7-4.3.2, "IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations," provides guidance for complying with requirements for safety systems that use digital computer systems. Other applicable regulatory guides are discussed in the acceptance criteria sections below.

Many standards exist that can be used to develop software for safety systems. The information in this BTP is generally based on the standards and reports referred to in Section C below, supplemented and modified as appropriate for attaining the required safety functions.

This BTP presents specific acceptance criteria for the elements of software reviews; however, important context information is found in the concepts contained in the referenced standards and reports. The reviewer should also understand the specific provisions of applicable regulatory guides.

3. Definitions

Activity group — A collection of software life cycle activities, all of which are related to a specific life-cycle topic. Eight activity groups are recognized in this BTP: planning, requirements, design, implementation, integration, validation, installation, and operations and maintenance (see Figure 7-A-1).

Design output — Documents, such as drawings and specifications, that define technical requirements of structures, systems, and components (ASME Std NQA-1, "Quality Assurance Requirements for Nuclear Facility Applications"). For software, design outputs are the products of the development process that describe the end product that will be installed in the plant. The design outputs of a software development process include software requirements specifications, software design specifications, hardware and software architecture designs, code listings, system build documents, installation configuration tables, operations manuals, maintenance manuals, and training manuals.

Deterministic timing — Timing is deterministic if the time delay between stimulus and response has a guaranteed maximum and minimum.

Documentation — Information recorded about a specific life cycle activity. Forty-one activities are recognized in this BTP (see Figure 7-A-1). Documentation includes software life-cycle design outputs and software life-cycle process documentation. A document may be in written or electronic format, and may contain text, illustrations, tables, computer files, program listings, binary images, and other forms of expression. A document for an activity may be packaged with documents for other activities, or documents for non-software life-cycle activities. A document for an activity may be divided into several individual entities.

The following paragraphs define the software planning characteristics important to safety system software. The definitions given are specific to software. The planning characteristics can be divided into three sets: management, implementation, and resource characteristics.

Planning Characteristics

- Management — Those characteristics of planning documents that are primarily significant to the managing of the project activities described in the planning document.

  - Purpose — A description of the reasons for the existence of the planning document, and the objectives which are to be satisfied by the planning document.

  - Organization — The organizational structure used to achieve the purpose of the planning document, including organizational boundaries and interfaces to other organizations.

  - Oversight — A specification of the methods used to oversee the work covered by the planning document.

  - Responsibilities — The duties of the organization covered by the planning document, and of the individuals within that organization.

  - Risks — The method used to identify, assess and manage risks that may interfere with achieving the purpose of the planning document.

  - Security — The methods used to protect the information created by or reviewed by the organization covered by the planning document from inadvertent or malicious alteration.

- Implementation — Those characteristics of planning documents that describe the work necessary to achieve the purpose of the planning documents.

  - Measurement — A set of indicators used to determine the success or failure of the activities and tasks defined in the planning document.

  - Procedures — The work necessary in order to achieve the purpose of the planning document,

  - Record keeping — Identification of the documentation required in order to demonstrate that the purpose of the planning document has been achieved, and the tasks necessary to store, handle, retain and ship that documentation have been accomplished.

  - Schedule — The time order of events necessary to achieve the purpose of the planning document, given either as absolute dates, ranges of dates, or offsets from other dates.

- Resources — The material resources necessary to carry out the work defined in the planning document.

  - Budget — The financial resources necessary to carry out the work.

  - Methods/tools — The methods and techniques by which the work will be carried out, and the tools used to implement those methods.

  - Personnel — The numbers, qualification, and training of personnel required to carry out the work defined in the planning document.

  - Standards — The international, national, industry and company standards and guidelines to be followed in the work defined in the planning document.

The following paragraphs define the software characteristics important to safety system software. The definitions given are specific to software. Software characteristics can be divided into two sets: functional characteristics and software development process characteristics. The first set includes those characteristics that directly relate to the actions that the safety system software must take, while the second includes those characteristics of the software development process that contribute to assurance that the software will perform the required actions. Both sets are important in safety system software. The sets, and the definitions of the characteristics, are listed below.

Functional Characteristics

- Accuracy — The degree of freedom from error of sensor and operator input, the degree of exactness exhibited by an approximation or measurement, and the degree of freedom from error of actuator output.

- Functionality — The operations which must be carried out by the software. Functions generally transform input information into output information in order to affect the reactor operation. Inputs may be obtained from sensors, operators, other equipment, or other software. Outputs may be directed to actuators, operators, other equipment, or other software.

- Reliability — The degree to which a software system or component operates without failure. This definition does not consider the consequences of failure, only the existence of failure.

- Robustness — The ability of a software system or component to function correctly in the presence of invalid inputs or stressful environmental conditions. This includes the ability to function correctly despite some violation of the assumptions in its specification.

- Safety — Those properties and characteristics of the software system that directly affect or interact with system safety considerations. The other characteristics discussed in this BTP are important contributors to the overall safety of the software-controlled safety system, but are primarily concerned with the internal operation of the software. The safety characteristic, however, is primarily concerned with the effect of the software on system hazards and the measures taken to control those hazards.

- Security — The ability to prevent unauthorized, undesired, and unsafe intrusions. Security is a safety concern insofar as such intrusions can affect the safety-related functions of the software.

- Timing — The ability of the software system to achieve its timing objectives within the hardware constraints imposed by the computing system being used.

Software Development Process Characteristics

- Completeness — Those attributes of the planning documents, implementation process documents and design outputs that provide full implementation of the functions required of the software. The functions which the software is required to perform are derived from the general functional requirements of the safety system, and the assignment of functional requirements to the software in the overall system design.

- Consistency — The degree of freedom from contradiction among the different documents and components of a software system. There are two aspects to consistency. Internal consistency denotes the consistency within the different parts of a component for example, a software design is internally consistent if no set of design elements are mutually contradictory. External consistency denotes the consistency between one component and another for example, software requirements and the resulting code are consistent with one another if there are no contradictions between the requirements and the code.

- Correctness — The degree to which a design output is free from faults in its specification, design, and implementation. There is considerable overlap between correctness properties and properties of other characteristics such as accuracy and completeness.

- Style — The form and structure of a planning document, implementation process document or design output. Document style refers to the structure and form of a document. This has connotations of understandability, readability, and modifiability. Programming style refers to the programming language characteristics of the software and programming techniques which are mandated, encouraged, discouraged, or prohibited in a given implementation.

- Traceability — The degree to which each element of one life cycle product can be traced forward to one or more elements of a successor life cycle product, and can be traced backwards to one or more elements of a predecessor life cycle product.

- Unambiguity — The degree to which each element of a product, and of all elements taken together, have only one interpretation.

- Verifiability — The degree to which a software planning document, implementation process document or design output is stated or provided in such a way as to facilitate the establishment of verification criteria and the performance of analyses, reviews, or tests to determine whether those criteria have been met.

4. Purpose

The purpose of this BTP is to provide guidance for NRC staff to verify conformance with the previously cited regulatory bases and standards in the design of digital computer systems. This BTP has three objectives:

- To confirm that plans exist that will provide a high-quality software life cycle process, and that these plans commit to documentation of life cycle activities that permit the NRC staff to evaluate the quality of the design features upon which the safety determination is based.

- To verify that implementation of the software life cycle process meets the criteria expected for high-quality software.

- To assess the adequacy of the design outputs.

B. Branch Technical Position

1. Introduction

Digital I&C safety systems must be designed, fabricated, installed, and tested to quality standards commensurate with the importance of the safety functions to be performed. Implementation of an acceptable software life cycle provides the necessary software quality.

Digital I&C systems may share code, data transmission, data, and process equipment to a greater degree than analog systems. Although this sharing is the basis for many of the advantages of digital systems, it also raises a key concern: a design using shared data or code has the potential to propagate a common-cause or common-mode failure via software errors, thus defeating the redundancy achieved by the hardware architectural structure. Greater sharing of process equipment among functions within a channel increases the consequences of the failure of a single hardware module and reduces the amount of diversity available within a single safety channel.

Because of these concerns, the Staff review of digital I&C systems emphasizes quality, defense-in-depth, and diversity as protection against common-mode failures within and between channels. Software quality is an important element in preventing the propagation of common-mode failures.

Commercial-off-the-shelf software and software embedded in commercial-off-the-shelf components, such as meters, circuit breakers, or alarm modules should be appropriately evaluated to confirm that required characteristics are met. EPRI Topical Report TR-106439, "Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications," describes an acceptable method for performing this evaluation. NUREG/CR-6421, "A Proposed Acceptance Process for Commercial Off-the-Shelf (COTS) Software in Reactor Applications," provides additional background information. The guidelines of this BTP may be used as appropriate in assessing the software engineering processes used to develop commercial software. See the discussion of the commercial dedication of predeveloped software (PDS) in Appendix 7.0-A.

The development of safety system software should progress according to a formally defined life cycle. Many life cycles have been defined in the technical literature and in national and international standards. These differ in the definitions of life cycle activity groups and in the order in which life cycle activities are performed. An appropriate set of life cycle activities is provided in Reg. Guide 1.173, "Developing Software Life Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std 1074, "Standard for Developing Life Cycle Processes." The software developer should select and document the software life cycle, and specify the products that will be produced by that life cycle. The software developer may be the applicant/licensee, the vendor, a company working on behalf of either, or a commercial software development company.

All software development life cycles share certain characteristics. The activities that will be performed can be grouped into a number of categories (termed activity groups here); the activity groups are common to all life cycles. Life cycle activities produce process documents and design outputs which can be reviewed and assessed. The documents to be provided for each life cycle activity group are shown in Figure 7-A-1. It is acceptable to package documents differently than shown in the figure. For example, information which is assumed to be provided in two or more documents could be combined by the software developer into a single document, and information which is assumed to be provided in a single document could be provided in two or more documents. Further information on life cycles, activity groups, and document contents can be found in NUREG/CR-6101.

2. Information to be Reviewed

The information to be reviewed is subdivided into three topic areas: software life cycle process planning, listed in Section 2.1; software life cycle process implementation, listed in Section 2.2; and software life cycle process design outputs, listed in Section 2.3.

The applicant/licensee need not develop a separate document for each of the topics identified below; however, project documentation should encompass all of the topics. The information reviewed need not be in separate documents. It is acceptable to package the information with other engineering project information, provided that the required information exists. This is particularly true of software life cycle planning information. For example, the software safety plan may be included in a general project safety plan.

2.1 Software Life Cycle Process Planning

The information to be reviewed is contained in the following documents:

- Software management plan.
- Software development plan.
- Software quality assurance plan.
- Integration plan.
- Installation plan.
- Maintenance plan.
- Training plan.
- Operations plan.
- Software safety plan.
- Software verification and validation plan.
- Software configuration management plan.

2.2 Software Life Cycle Process Implementation

The information to be reviewed is contained in the following:

- Safety analyses.
- Verification and validation analysis and test reports.
- Configuration management reports.

One or more sets of these reports should be available for each of the following activity groups:

- Requirements.
- Design.
- Implementation.
- Integration.
- Validation.
- Installation.
- Operations and maintenance.

2.3 Software Life Cycle Process Design Outputs

The information to be reviewed is contained in the following:

- Software requirements specifications (SRS).
- Hardware and software architecture descriptions (SAD).
- Software design specifications (SDS).
- Code listings.
- Build documents.
- Installation configuration tables.
- Operations manuals.
- Maintenance manuals.
- Training manuals.

System requirements documents should also be examined to provide context for this review.

3. Acceptance Criteria

The acceptance criteria are subdivided into three areas matching the information to be reviewed listed in Section 2: software life cycle process planning criteria, enumerated in Section 3.1; software life cycle process implementation criteria, enumerated in Section 3.2; and software life cycle process design output criteria, listed in Section 3.3. The topic areas and documentation groups arise naturally from a disciplined engineering process that has three major stages: planning, design process implementation, and design output.

3.1 Acceptance Criteria for Software Life Cycle Process Planning

This section addresses acceptance criteria for planning activities. The acceptance criteria address specific software development planning activities and products. These products, when found to be acceptable, provide the reviewer with additional criteria for reviewing the processes and products of subsequent life cycle activities, as discussed in Sections 3.2 and 3.3 below.

Acceptance criteria are divided into three sets: management characteristics, implementation characteristics, and resource characteristics. Each of these is further divided into specific characteristics, as shown in the following table. Not all specific characteristics occur for every plan.

Management Characteristics	Implementation Characteristics	Resource Characteristics
Purpose	Measurement	Budget
Organization	Procedures	Methods/tools
Oversight	Record keeping	Personnel
Responsibilities	Schedule	Standards
Risks		
Security		

Software development process characteristics are defined in Section A.3 above. All planning documents should be evaluated for the following process characteristics: consistency, style, traceability, unambiguity and verifiability. Each plan should be internally consistent, and the complete set of plans should be mutually consistent. Plans should be documented so that they can be understood both by the users of the plan and by the reviewers. The software management plan should be traceable back to system management planning; the remaining software plans should be traceable back to the software management plan; and the various process implementation documents and the design outputs should be traceable back to the relevant plans. The set of plans should not be ambiguous. It should be possible to verify that the plans have been followed during the software project. The review and assessment of the quality of the plans provide a means of judging the competency of the development organization and management.

It may be the case, particularly when the applicant/licensee is planning for future plants, that the software plans are created in stages as information becomes available. For example, budget and schedule information may not be available when the initial plans are created. This is acceptable, provided that the information is added to the plans prior to the time the information is needed to carry out the plans.

a. Software Management Plan

The software management plan describes the management aspects of the software development project. It may be part of a general company software management plan, a project engineering management plan, or may be split among various management plans and company procedures. The software management plan should exhibit the management, implementation and resource characteristics listed below.

Management Characteristics

The management characteristics that the software management plan should exhibit include purpose, organization, oversight, responsibilities, and security.

Purpose requires that the intent of the software project be defined in the software management plan. The plan should list the general functions the software will be expected to provide, and should provide an overview of the system within which the software will reside. A general overview of the project should be provided. The assumptions upon which the project is based should be stated. The scope of work for the software project, and the product and process goals, should be discussed.

Organization requires a description of the software project planning organization. The plan should describe the software project organizational structure, and should describe the interfaces and boundaries between the project organization and other company organizations. Management reporting channels should be described. The methods by which subcontractors and suppliers will be managed should be described.

The plan should ensure that the quality assurance organization, the software safety organization and the software verification and validation (V&V) organization maintain independence from the development organization. In particular, the plan should ensure that these assurance organizations not report to the development organization, and not be subject to the financial control of the development organization.

Oversight requires that the strategy for managing the software project be specified. Project priorities should be listed. A method should be described to monitor progress against the software management plan and to document progress at regular intervals in progress reports. A method should exist to identify any deviations from the software management plan in time to take corrective action.

Responsibilities requires a definition of the duties of each member of the project’s management and technical teams. The plan should include a policy statement that the development personnel who produce each design output required by the software development plan have the primary responsibility for the quality of that output.

Security requires a description of the methods to be used to prevent contamination of the developed software by viruses, Trojan horses or other nefarious intrusions. The required security level for each project phase should be given.

Implementation Characteristics

The implementation characteristics that the software management plan should exhibit include measurement and procedures.

Measurement requires the definition of a set of management indicators which will be used to monitor and control the project. The plan should require that data associated with project management be systematically collected and analyzed to determine the effectiveness of project management.

Procedures requires a description of the process by which the project will be managed. The plan should describe project priorities, project assumptions, and monitoring and control methods. It should describe the approach to be followed for recording the rationale for key decisions made in specifying, designing, implementing, procuring and assessing the software. A list of all deliverable software, test software, support software and associated documentation should be included. Project management reviews should be specified. The means for performing corrective action and process improvement should be described. Management reports should be described, and reporting channels should be described. Periodic progress reports should be required.

The software management plan should define the means by which the remaining plans will be produced. It should provide a means of managing externally and internally generated changes in any of the plans. The people responsible for reviewing the various project plans and any changes to those plans should be listed, by name or by title. A means should exist for generating changes to the plans and for evaluating suggested changes. The plans should be under configuration management control.

Resource Characteristics

The resource characteristics that the software management plan should exhibit include budget, methods/tools and personnel.

Budget requires a project budget for all project activities. A means should exist to track and report resource expenditures. Sufficient resources should exist to carry out the defined tasks. The plan should ensure that quality assurance budgets, safety budgets, and V&V budgets not be subject to expropriation by the software development organization, in order to maintain financial independence of these assurance activities.

Methods/tools requires a description of the means used to manage the project. The plan should identify the methods, techniques and tools required to carry out the project management, including office equipment, computer hardware, and computer software.

Personnel requires a specification of the numbers, qualifications, training and types of personnel required to conduct the project. Personnel resources for each project phase should be listed.

Safety and V&V personnel should be competent in software engineering in order to ensure that software safety and software V&V are effectively implemented.

b. Software Development Plan

The software development plan describes the plan for technical project development. It may be part of a general company software management plan, may be project specific, or may be split among various management plans. The software development plan should exhibit the management, implementation and resource characteristics listed below.

Management Characteristics

The management characteristics that the software development plan should exhibit include purpose, organization, oversight, and risks.

Purpose requires a description of the objectives of each life cycle phase and its context within the overall project.

Organization describes the software life cycle that will be used in the project. The life cycle should include uniquely identifiable development, verification and support processes with well-defined inputs and outputs. The life cycle model should be documented in the plan. Reg. Guide 1.173, "Development Software Life Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std. 1074, "IEEE Standard for Developing Life Cycle Processes," describes acceptable methods of organizing the software life cycle.

Oversight requires that the strategy for managing the technical development effort be specified. Project priorities should be listed. Required software quality factors should be identified and ordered by importance. A method should be provided to monitor progress against the software development plan and to document progress at regular intervals in progress reports. A method should exist to identify any deviations from the software development plan in time to take corrective action.

Risks requires that project risks be identified, assessed and managed. The plan should describe the method to be used for risk identification, assessment and management, with particular attention to risks that have the potential for compromising safety. The plan should describe the method to be used to identify and assess the risk factors associated with product engineering, development environment and program constraints. It should describe the mechanisms for tracking the risk factors and implementing contingency plans. Risk factors that should be included include system risks, mechanical/electrical hardware integration, risks due to size and complexity of the product, the use of predeveloped software, cost and schedule, technological risk, and risks from program interfaces (maintenance, user, associate contractors, subcontractors, etc.). The plan should identify key design and implementation issues, and the preliminary studies, simulation modeling, and prototyping required to resolve them.

Implementation Characteristics

The implementation characteristics that the software development plan should exhibit include measurement, procedures, and schedule.

Measurement requires a set of indicators used to determine the success or failure of the technical aspects of the development process and the resulting design outputs. The plan should require data associated with the technical development of the design outputs to be collected and analyzed to determine software quality. The error rate found during the development phases should be measured, recorded, analyzed and reported.

Procedures requires the division of each life cycle activity into well-defined tasks. The inputs to each activity and each task should be provided, and the sources of those inputs should be identified. The conditions that must be satisfied before each activity can begin should be described. The outputs from each activity and each task should be provided, and the destination of those outputs should be identified. The plan should include a review at the end of each life cycle activity. Reports on the technical development work should be described. Reg. Guide 1.173 describes acceptable methods for defining the inputs and outputs of the life cycle activities.

Schedule requires a project schedule. The plan should identify key work packages, milestones and hold points. Sufficient intermediate milestones should be identified to avoid unexpected schedule delays. Reviews and audits should be included in the schedule as project milestones. The schedule should justify the time anticipated to complete each task. A single schedule that includes both management and technical activities is acceptable.

Resource Characteristics

The resource characteristics that the software development plan should exhibit include methods/tools and standards.

Methods/tools requires a description of the software development methods, techniques and tools to be used. The approach to be followed for reusing software should be described. The plan should identify suitable facilities, tools and aids to facilitate the production, management and publication of appropriate and consistent documentation and for the development of the software. It should describe the software development environment, including software design aids, compilers, loaders, and subroutine libraries. The plan should require that tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software which is to be developed using the tools. Methods, techniques and tools that produce results that cannot be verified to an acceptable degree or that are not compatible with safety requirements should be prohibited, unless analysis shows that the alternative would be less safe.

Standards requires a list the international, national, industry, and company standards and guidelines (including Reg. Guides) to be followed in the project. This should include software requirements standards, software design standards and software coding standards and internal standards and engineering and physical standards that form the basis for the plant safety analysis. The Reg. Guides listed in Table 7–1 should be considered for inclusion in the list of standards.

c. Software Quality Assurance Plan

The software quality assurance (QA) plan should exhibit the management, implementation and resource characteristics listed below. It is acceptable to include the software QA plan in a more general project QA plan if the required content exists. The software QA plan should conform with the requirements of 10 CFR 50, Appendix B, and the applicant/licensee's overall quality assurance program.

Management Characteristics

The management characteristics that the software quality assurance plan should exhibit include purpose, organization, and responsibilities.

Purpose requires a general description of the quality assurance process, and the goals of that process. The plan should list the general functions the software QA organization will be expected to perform and specific objectives for this project, if applicable.

Organization requires a description of the software QA organization. The plan should describe the boundaries between the software QA organization and other company organizations. Reporting channels should be described.

Responsibilities requires a definition of the responsibilities and authority of the software QA organization. The plan should require the software QA organization to assess and evaluate system safety, reliability and maintainability characteristics of the software.

Implementation Characteristics

The implementation characteristics that the software quality assurance plan should exhibit include measurement, procedures, and record keeping.

Measurement requires a set of indicators used to determine the success or failure of the software QA effort. The plan should require quality assurance data to be systematically collected and analyzed to determine software quality.

Procedures requires a description of the software QA procedures for the entire software life cycle. The plan should provide for QA participation in the assessment and review of project-specific standards, methods and tools. The plan should describe the methods, procedures and controls used to ensure that technical, quality and other requirements are accurately stated in project documentation. Procedures should exist to identify, track and resolve project conditions adverse to quality. The plan should ensure that traceability is maintained through all phases of the software life cycle. Required software quality factors (listed in Section B.3.3 below) should be identified. Software QA reports should be described.

The software QA organization should participate in formal reviews and audits of the software development activity. Required reviews and audits should be listed in the plan, including review documentation requirements, evaluation criteria, anomaly reporting and anomaly resolution procedures. Reg. Guide 1.168, "Verification, Validation, Reviews and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std 1028, "IEEE Standard for Software Reviews and Audits," describes acceptable methods for QA software reviews and audits.

Record keeping requires a description of the software QA record keeping requirements and procedures. A list of the documents subject to software quality assurance oversight should be included. The plan should describe storage, handling, retention and shipping procedures for these documents and for project quality records. Document structures (such as an annotated table of contents) should be provided. The document control mechanism should be specified.

Resource Characteristics

The resource characteristics that the software quality assurance plan should exhibit include methods/tools and standards.

Methods/tools requires a description of the means that will be used to accomplish the quality assurance function. The plan should identify suitable facilities, equipment, methods, techniques and tools to facilitate the performance of the QA work. Computer equipment and software used to perform the QA work should be specified.

Standards requires a method to ensure that approved standards, methods and tools are applied throughout the software life cycle. The plan should provide a method to establish and maintain the standards and methods for software QA, software V&V and software configuration management (CM).

d. Software Integration Plan

The software integration plan should exhibit the management, implementation, and resource characteristics listed below.

Management Characteristics

The management characteristics that the software integration plan should exhibit include purpose, organization and responsibilities.

Purpose requires a general description of the software integration process, the hardware/software integration process and the goals of those processes. The plan should include a general description of the software integration process and of the hardware/software integration process.

Organization requires a description of the software integration organization. The plan should describe the boundaries between the software integration organization and other company organizations. Reporting channels should be described. It is acceptable for the integration organization to report to the development organization, or to be part of the development organization.

Responsibilities requires a definition of the responsibilities and authority of the software integration organization.

Implementation Characteristics

The implementation characteristics that the software integration plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the integration effort. The plan should require that data associated with the integration of the software, and of the hardware/software combination, be collected and analyzed to determine the adequacy of the integration effort. The error rate found during integration activities should be measured, recorded, analyzed and reported.

Procedures requires an integration strategy. The plan should include methods, procedures and controls for software integration, and for combined hardware/software integration. Integration design outputs and reports should be described. The plan should require documentation describing the software integration tests to be performed, the hardware/software integration tests to be performed, and the expected results of those tests.

Resource Characteristics

The resource characteristics that the software integration plan should exhibit include methods/tools.

Methods/tools requires a description of the methods, techniques and tools that will be used to accomplish the integration function. The plan should require that integration tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software which is to be created using the tools.

e. Software Installation Plan

The software installation plan should exhibit the management, implementation, and resource characteristics listed below.

Management Characteristics

The management characteristics that the software installation plan should exhibit include purpose, organization and responsibilities.

Purpose requires a general description of the installation process, and the goals of that process. A general description of the environment (such as temperature, humidity, vibration, and rack space) within which the computer system and software system is qualified to operate should be included in the plan.

Organization requires a description of the software installation organization. The plan should describe the boundaries between the software installation organization and the broader safety system installation organization. Reporting channels should be described. It is acceptable for the installation to be performed by the development organization or by the customer.

Responsibilities requires a definition of the responsibilities and authority of the software installation organization. If installation is performed by the customer, then the delineation of responsibility between the development organization and the customer should be defined in such a way that misunderstandings in communications between the two organizations are kept to a minimum.

Implementation Characteristics

The implementation characteristics that the software installation plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the installation effort. The plan should require that data associated with the installation be collected and analyzed. The error rate found during installation activities should be measured, recorded, analyzed and reported.

Procedures requires a description of the installation strategy. The plan should describe procedures for software installation, and for combined hardware/software installation. The plan should describe the methods, procedures and controls used to ensure that the success or failure of the installation effort can be reliably determined. Checks should be required to ensure that the computer system is functional, that the sensors and actuators are functional, that all cards are present and installed in the correct slots, and that the communication system is correctly installed. A check should be required to ensure that the correct software versions are installed on the correct computers. Installation reports should be described. The plan should require that anomalies discovered during installation be reported to the developer and resolved prior to placing the software into operation. Either this plan, or the software V&V plan, should require adequate testing to provide confidence that the installed system will perform its safety function.

Plans for installation of software on installed systems in operating plants should recognize the need to declare all affected functions inoperable according to the plant's technical specifications before proceeding with installation, and to conduct appropriate return-to-service testing before declaring the modified function operable.

Resource Characteristics

The resource characteristics that the software installation plan should exhibit include methods/tools.

Methods/tools requires a description of the methods, techniques and tools that will be used to accomplish the installation function. The plan should require that installation tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software which is to be installed using the tools.

f. Software Maintenance Plan

The software maintenance plan should exhibit the management, implementation and resource characteristics listed below.

Management Characteristics

The management characteristics that the software maintenance plan should exhibit include purpose, organization, responsibilities, risks, and security.

Purpose requires a general description of the software maintenance process and the goals of that process. The plan should list the general functions that the software maintenance organization will be expected to perform, and provide general information on obtaining field trouble reports. Maintenance should be limited to the process of modifying a software design output to repair nonconforming items or to implement pre-planned actions necessary to maintain performance. Modifications to improve performance or other attributes, or to adapt the design outputs to a modified environment should be considered design changes.

Organization requires a description of the software maintenance organization. The plan should describe the boundaries between the software maintenance organization and other company organizations. Reporting channels should be described. Formal communication channels between the maintenance organization and the customers using the software should be provided, so that incorrect behavior of the software during operation can be identified, isolated and corrected. This communication structure should provide assurance that software failures during operation will not be ignored.

Responsibilities requires a definition of the responsibilities and authority of the software maintenance organization.

Risks requires a description of the method used for software risk management during maintenance, with particular attention to risks that have the potential for compromising safety.

Security requires a description of the methods to be used to prevent contamination of the corrected software by viruses, Trojan horses or other nefarious additions. The required security level for each maintenance project phase should be provided.

The plan should identify the controls needed over maintenance activities and maintenance and test equipment to prevent unauthorized changes to hardware, software and system parameters. At a minimum, the potential for introducing unauthorized changes during repair, testing and calibration should be addressed.

Implementation Characteristics

The implementation characteristics that the software maintenance plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the maintenance effort. The plan should require that data associated with maintenance activities be collected and analyzed to determine the effectiveness of the maintenance effort. The error rate found during maintenance activities should be measured, recorded, analyzed and reported.

Procedures requires a description of the maintenance strategy. The plan should include procedures for problem reporting by customers, and for resolution of those problem reports. The problem reporting procedure should give time and date of occurrence, a brief description of the problem (including the state of the system at the beginning of the occurrence) and a description as to what was done to correct the problem. It should require that reported problems be evaluated to allow the identification of nonconforming items and the performance of corrective actions as described in Sections XV and XVI of 10 CFR 50 Appendix B. The plan should describe the process for identification, documentation, evaluation, segregation where practical, and disposition of nonconforming items, and for notification to affected organizations. Evaluation of nonconforming items and corrective actions should include as appropriate evaluation with respect to the requirements of 10 CFR 50.59 and reporting per the requirements of 10 CFR 21. Nonconformances to design requirements dispositioned "use-as-is" or "repair" should be subject to design control (including verification and validation, quality assurance, safety analysis, and configuration management) measures commensurate with those applied to the original design. The plan should require that as-built records reflect any accepted deviations and justification for that acceptance.

Because any error in safety system software presents the potential for common-mode failure of redundant functions, the maintenance plan should require timely evaluation of the effects of reported problems to support equipment operability determinations as required by plant technical specifications.

Periodic analysis and reporting of problems and their resolution should be required along with recommendations for improving operation. There should be a requirement for reporting what actions were taken regarding these recommendations.

Resource Characteristics

The resource characteristics that the software maintenance plan should exhibit include methods/tools.

Methods/tools requires a description of the methods, techniques and tools that will be used to accomplish the maintenance function. The plan should describe the facilities required to maintain the delivered software. It should list and describe the software, hardware and associated documentation required to maintain the delivered software. The plan should require that maintenance tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software which is to be created using the tools.

g. Software Training Plan

The software training plan should exhibit the management, implementation, and resource characteristics listed below.

Management Characteristics

The management characteristics that the software training plan should exhibit include purpose, organization, and responsibilities.

Purpose requires a description of the means necessary to ensure that training needs of appropriate plant staff, including operators and I&C engineers and technicians, are fully achieved. The plan should include a general description of the training facilities.

Organization requires a description of the software training organization. The interfaces between the training organization and the project management organization should be described. Reporting channels should be described. Trainers should have the necessary knowledge of the software operation to ensure that trainees understand its operating and maintenance requirements.

Responsibilities requires a definition of the responsibilities and authority of the training organization and training by customers.

Implementation Characteristics

The implementation characteristics that the software training plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the training effort. The plan should require that training data be collected and analyzed to determine the effectiveness of the training effort. The trainee error rate found at the end of training activities should be measured, recorded, analyzed and reported.

Procedures requires a description of the training procedures. The plan should list any documentation required to support the training effort. The training program should be described. The plan should require that training be specific to different job functions. Training products and reports should be described. Reporting requirements should be specified.

Resource Characteristics

The resource characteristics that the software training plan should exhibit include methods/tools.

Methods/tools requires a description of the methods, techniques and tools that will be used to accomplish the training function. Training should be carried out on a training system which is equivalent to the actual hardware/software system.

h. Software Operations Plan

The software operations plan should exhibit the management, implementation and resource characteristics listed below.

Management Characteristics

The management characteristics that the software operations plan should exhibit include purpose, organization, responsibilities and security.

Purpose requires a general description of the operation of the software. The plan should include a general description of the functions that the software is to perform, and a general discussion of the means of carrying out those functions.

Organization requires a description of the organizational structure necessary to control the software operation. The plan should specify operator interface stations and actions required to support operation.

Responsibilities requires a description of the responsibilities and authority of the operators.

Security requires a description of the security requirements for operating the software system. The operations plan should identify the controls needed over operation activities to prevent unauthorized changes to hardware, software and system parameters, the monitoring activities needed to detect penetration or attempted penetration of the system, and contingency plans needed to ensure appropriate response to penetration.

Implementation Characteristics

The implementation characteristics that the software operations plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the operating procedures. The error rate found during operation activities should be measured, recorded, analyzed and reported.

Procedures requires a description of the procedures necessary to start, operate and stop the software system. The plan should require a description of procedures for executing the software in all operating modes, and procedures for ensuring that the software state is consistent with the plant operating mode at all times. The plan should require a description of backup procedures for data and code, and the intervals at which backup should occur. The plan should require a list of error messages, giving a description of the error indication, the probable interpretation of the error indication, and steps to be taken to resolve the situation.

Resource Characteristics

The resource characteristics that the software operations plan should exhibit include methods/tools.

Methods/tools requires a description of the methods, techniques and tools that will be used to operate the software system. The plan should describe the facilities required to operate the delivered software. It should list and describe the software, hardware and associated documentation required to operate the delivered software.

i. Software Safety Plan

The software safety plan should exhibit the management, implementation and resource characteristics listed below. It is acceptable to include the software safety plan in a more general project safety plan if the required content exists. The software safety plan should conform with the requirements of the design basis for the software applications involved.

Management Characteristics

The management characteristics that the software safety plan should exhibit include purpose, organization, responsibilities and risks.

Purpose requires a specification of the purpose and scope of the software safety activities. The plan should provide a general description of the software safety effort, and the intended interactions between the software safety organization and the general system safety organization.

Organization requires a description of the software safety organization. The plan should describe the boundaries and interfaces between the software safety organization and other company organizations. It should show how the software safety activities are integrated with the system safety activities, how the software safety activities are coordinated with the development activities, and the interactions between the software safety organization and the software V&V organization. The plan should designate a single safety officer that has clear responsibility for the safety qualities of the software being constructed.

Responsibilities requires a definition of the responsibilities and authority of the software safety organization. The plan should specify the person or group responsible for each software safety task. A designated safety officer should have clear authority for enforcing safety requirements in the software requirements specification, the design, and the implementation of the software. The safety officer should have the authority to reject the use of predeveloped software if the software cannot be shown to be adequately safe or if, in using a tool, it cannot be shown that the tool will not impact the safety of the final software system. The plan should require that safety personnel be aware of the safety implications of hardware, software and interfaces between them.

Risks requires a description of the methods to be used to reduce safety risks caused by software failures to an acceptable level. The plan should describe the method to be used to ensure that hazards which software is expected to control are resolved in an acceptable manner. The plan should include a requirement that a safety analysis be performed and documented on each of the principal design documents: requirements, design descriptions, and source code. Hazards, including abnormal events and conditions and malicious modifications, should be analyzed and documented. Hazard reduction efforts should be documented.

Implementation Characteristics

The implementation characteristics that the software safety plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the software safety effort. The plan should require that software safety data be systematically collected and analyzed to determine the effectiveness of the software safety effort.

Procedures requires a description of the software safety strategy. The plan should describe the management of the software safety activities within the development organization. It should provide procedures for resolving safety issues. The plan should require that problems encountered in implementing the safety program be brought to the attention of the project manager. A procedure should exist for assuring resolution of identified unacceptable risks. The plan should describe methods to be used to implement each safety task. A method should exist to identify hazards caused by software, and to identify hazards whose resolution will be under the control of software.

The plan should require that appropriate safety requirements be included in the software requirements specification. It should define the safety-related activities to be carried out for each set of life cycle activities, from requirements through operation and maintenance. The plan should identify all documentation required for the proper and safe operation of the software. Procedures should require monitoring the software safety function performance during operation of the system.

The plan should require that hazards identified by plant safety analysis, system safety analysis and security vulnerability assessment be traceable to the software safety analysis whenever these hazards can affect software operability or whenever software has a role in controlling the hazard.

Resource Characteristics

The resource characteristics that the software safety plan should exhibit include methods/tools and standards.

Methods/tools requires a description of the methods and tools used to carry out the safety activities. The plan should specify a process for selecting tools. It should describe a method for preventing the inadvertent introduction of hazards by the use of project tools.

Standards requires a list of the international, national, industry and company standards and guidelines to be followed by the safety organization.

j. Software Verification and Validation Plan

The software verification and validation (V&V) plan should exhibit the management, implementation and resource characteristics listed below. Reg. Guide 1.168, which endorses ANSI/IEEE Std 1012, "Software Verification and Validation Plans," shows an acceptable organization and content for this plan.

Management Characteristics

The management characteristics that the software V&V plan should exhibit include organization, oversight, responsibilities, and risks.

Purpose requires a definition of the purpose and scope of the software V&V activities. The plan should include a general description of the software V&V process.

Organization requires a description of the V&V organization. The plan should describe the boundaries and interfaces between the V&V organization and other company organizations. Reporting channels should be described. The relationship among the different V&V tasks should be specified. The plan should require that the V&V organization be independent of the development organization. It should require that formal communication between the V&V and design organizations be documented.

Responsibilities requires a definition of the responsibilities and authority of the software V&V organization. The plan should specify the person or group responsible for the successful completion of each V&V task. It should specify the person with authority to approve the successful completion of each V&V task. It should specify the person with authority to approve the release of the reviewed and tested software design outputs.

Risks requires a specification of the methods used to identify and manage risks associated with the V&V process. The plan should specify a method for evaluating the risk to safety associated with each software item. It should describe a method for identifying the risk associated with each V&V task. A contingency plan should be included to identify risk factors that may cause the V&V task to fail to perform its functions, and to recover from any such failure.

Implementation Characteristics

The implementation characteristics that the software V&V plan should exhibit include measurement and procedures.

Measurement requires a set of indicators used to determine the success or failure of the software V&V effort. The plan should specify the criteria to be used to verify the completion of each V&V task. Evaluation criteria should be provided for test plans, test specifications, test procedures and test cases. Evaluation criteria should be provided for review plans, review specifications and review procedures. The plan should require that V&V analysis, review and testing data be systematically collected and analyzed to determine the effectiveness of the V&V effort. The error rate found during software reviews and software testing should be measured, recorded, analyzed and reported.

Procedures requires a description of the software review and testing strategy. The plan should describe the management of the software V&V activities. It should specify the V&V tasks which will be carried out, including the planning assumptions for each task. It should establish the procedures and methods by which each V&V task will be performed, including the activities required to evaluate each software design output and each development activity in order to demonstrate that the system and software requirements have been met. It should establish procedures to ensure that systems in which errors are detected are appropriately analyzed, reported, corrected and reassessed. The plan should provide procedures for evaluating the risks associated with each project development activity. It should include a procedure for evaluating the effect of proposed software changes on planned reviews and tests.

Anomaly reports should be generated and disseminated. A method should be specified for resolving discrepancies identified during the verification of each V&V task. Procedures should be specified for selecting test cases, and for software review activities.

The plan should describe V&V reporting requirements. It should require that reports document all V&V activities, including the personnel conducting the activities, procedures and results. This includes review documentation requirements, evaluation criteria, error reporting, and anomaly resolution procedures. V&V reports should summarize the positive practices and findings as well as negative practices and findings. The reports should summarize the actions performed and the methods and tools used.

The plan should include a description of all required testing plans, specifications, procedures and cases. This includes unit testing, integration (subsystem) testing, system validation testing, installation (acceptance) testing, and the regression testing of modifications. The description should also include test documentation requirements, readiness and evaluation criteria, error reporting, and anomaly resolution procedures. Testing documentation should include test item descriptions, test data, test logs, the identities of testers, types of observations, results and acceptability, and actions taken in connection with any deficiencies. Test case documentation should specify expected results and actual results. Reg. Guide 1.170, "Software Test Documentation for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std 829, "IEEE Standard for Software Test Documentation," describes acceptable methods for documenting test plans, test specifications, test procedures, test cases, and test reports. Reg. Guide 1.171, "Software Unit Testing for Digital Computer Software Used in Safety

Systems of Nuclear Power Plants," which endorses ANSI/IEEE Std 1008, "IEEE Standard for Software Unit Testing," describes acceptable methods for performing unit tests.

Resource Characteristics

The resource characteristics that the software V&V plan should exhibit include methods/tools and standards.

Methods/tools requires a description of the methods, equipment, instrumentation and tools used to carry out each V&V task. Test methods should be specified for unit, integration, validation, installation and regression testing. The plan should specify a process for selecting tools. The hardware and software environment within which the V&V tools are to be applied and any necessary controls should be described.

Standards requires a list of the international, national, industry and company standards and guidelines to be followed by the V&V organization.

k. Software Configuration Management Plan

The software configuration management (CM) plan should exhibit the management, implementation and resource characteristics listed below. Reg. Guide 1.169, "Configuration Management Plans for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std. 828, "IEEE Standard for Software Configuration Management Plans," provides an acceptable organization for this plan. It is acceptable for the software configuration management plan to be included in a more general system configuration management plan if the required content exists.

Management Characteristics

The management characteristics that the software CM plan should exhibit include purpose, organization, and responsibilities.

Purpose requires a definition of the purpose and scope of the software CM activities. The plan should list the general functions the software CM organization will be expected to perform.

Organization requires a description of the software CM organization. The plan should describe the boundaries and interfaces between the CM organization and other company organizations. Reporting channels should be described. It is acceptable for the software configuration management organization to be a part of, or to report to, the development organization or a project CM organization.

Responsibilities requires a definition of the responsibilities and authority of the software CM organization. The plan should specify the person or group responsible for the successful completion of each CM task. It should define the duties of the configuration control board. It should specify the person who has the authority to release any software, data or documents for revision, and the person who has the authority to release any software, data or documents for operation after revision has been completed.

Implementation Characteristics

The implementation characteristics that the software CM plan should exhibit include measurement, procedures, and record keeping.

Measurement requires a set of indicators used to determine the success or failure of the software CM activity. The plan should require that data associated with configuration management be systematically collected and analyzed to determine the effectiveness of the CM effort. The plan should specify the criteria to be used to verify the completion of each CM task.

Procedures requires a description of the software configuration management strategy. The plan should specify procedures for identifying and naming configuration items. It should specify procedures for placing items under configuration control. It should describe the method for keeping data files and tables synchronized with the software that uses them, and for keeping software and its associated documentation synchronized. It should specify the procedure for associating source code with the derived object code and executable modules. Procedures should exist for managing software libraries. The plan should ensure the control and retrieval of qualification information associated with the software designs and code, software confirmation audits, and status accounting.

Items to be controlled should include: software requirements, designs, and code; support software used in development (exact versions); libraries of software components essential to safety; software plans that could affect quality; test software requirements, designs, or code used in testing; test results used to qualify software; analyses and results used to qualify software; software documentation; databases and software configuration data; predeveloped software items that are safety system software; software change documentation; and tools used in the software project for management, development or assurance tasks.

The plan should specify procedures for tracking problem reports, and for ensuring that each problem reported has been correctly resolved. The plan should describe the information required to approve a change request, and should ensure control of all software design changes. The relationship of software CM to other change control procedures, such as V&V anomaly handling and maintenance, should be described.

The plan should require periodic reviews and audits of the configuration baseline, including physical audits of the baseline.

The plan should include a description of the process used to maintain and track purchased items, such as software tools used to make the final product. A qualification procedure should be provided, and a method of tracking tool history, buglists, and errata sheets should enable the applicant/licensee to track which design outputs may be affected by discovered tool or purchased item deficiencies. The plan should describe procedures to control vendors supplying safety system software.

Record keeping requires a description of the software CM record keeping requirements. The plan should identify required CM records. Record structures (such as an annotated table of contents) should be provided. Procedures should exist for protecting configuration items. The plan should describe how configuration items will be stored, handled, retained and shipped. A tracking system should exist for managing configuration items, so that the revision history of each configuration item may be retrieved, and so that the latest revision of each configuration item may be easily identified. Procedures should exist for backup and disaster recovery.

Resource Characteristics

The resource characteristics that the software CM plan should exhibit include methods/tools and standards.

Methods/tools requires a description of the means that will be used to carry out each CM task. The plan should identify suitable facilities, methods, techniques and tools to facilitate the performance of the CM work. The plan should specify a process for selecting configuration management tools. The hardware and software environment within which the CM tools are to be applied and any necessary controls should be described.

Standards requires a list of the international, national, industry and company standards and guidelines to be followed by the software CM organization.

3.2. Acceptance Criteria for Software Life Cycle Process Implementation

This section addresses acceptance criteria for implementation activities. The acceptance criteria address specific software life cycle process implementation activities and documentation. These activities and products, when found to be acceptable, provide the reviewer with confidence that the plans listed in Section 2.1 above have been carried out.

The NRC staff reviewer confirms that the plans described in Section 3.1 have been followed by the software developer. The detailed acceptance criteria are provided by the software developer and evaluated by the NRC staff in its acceptance of the plans. In addition to verifying that plans have been followed, the reviewer should pay particular attention to the three areas discussed below. These activities are depicted in Figure 7-A-1 as process implementation.

a. Safety Analysis Activities

The software safety plan describes the safety analysis implementation tasks that are to be carried out by the applicant/licensee. The acceptance criterion for software safety analysis implementation is that the tasks in that plan have been carried out in their entirety. Documentation should exist that shows that the safety analysis activities have been successfully accomplished for each life cycle activity group. In particular, the documentation should show that the system safety requirements have been adequately addressed for each activity group; that no new hazards have been introduced; that the software requirements, design elements, and code elements that can affect safety have been identified; and that all other software requirements, design, and code elements will not adversely affect safety.

b. Software Verification and Validation Activities

The software V&V plan describes the V&V implementation tasks that are to be carried out by the applicant/licensee. The acceptance criterion for software V&V implementation is that the tasks in the plans have been carried out in their entirety. Documentation should exist that shows that the V&V tasks have been successfully accomplished for each life cycle activity group. In particular, the documentation should show that the requirements, design, code, integration, and installation design outputs satisfy the appropriate software development functional and process characteristics (as described in Section B.3.3 below).

Problems identified by the verification effort should be documented, together with any action items required to mitigate or eliminate each problem. A record should be kept of actions taken in response to the action items and the appropriate CM activities should be performed.

As part of the software V&V effort, a traceability matrix should be produced. This traceability matrix should clearly show the linkage between each requirement imposed on the software by the system requirements document and system design documents, and one or more requirements in the SRS. The matrix should allow traceability in both directions. It should be organized so that as design, implementation, and validation take place, traceability information can be added for these activities. It should be updated at the completion of each life cycle activity group. The final matrix should permit tracing from the system requirements and design through the software requirements, design, implementation, integration, validation, and installation.

The integration V&V activities should demonstrate that all unit and subsystem tests required by the V&V plan were successfully completed. Any anomalies or errors found during the tests should be resolved and documented. Final integration tests should be completed and documented. Reports should be written for each test run. These reports should include any anomalies found and actions recommended. The final integration V&V report should describe the procedures followed and the tests performed during integration. This report should be consistent with the integration plan.

The software validation activities should demonstrate that all validation tests required by the V&V plan were successfully completed. The testing process should contain one or more tests for each requirement in the SRS, as well as the acceptance criteria for each test. The result of each test should clearly show that the associated requirement has been met. Each test procedure should contain detailed information for the test setup, input data requirements, output data expectations, and completion time. Documentation should be produced for each test. Procedures should be included for handling errors and anomalies that are encountered during the testing. These procedures should include correction procedures (including configuration management), and provision for re-test until such time as the problems are resolved. A final report summarizing the validation testing should be provided. The report should contain a summary of problems and errors encountered during testing, and the actions taken to correct the problems encountered. The report should contain a statement that the validation testing was successful and that the software tested met all of the requirements of the SRS.

The installation (acceptance) test activities should document the test configuration, the required inputs, expected outputs, the steps necessary to execute the test, and the acceptance criteria for each test. The procedure should require that problems identified during the test activity, and any action items required to mitigate or eliminate each problem, be documented. Installation problems and their resolution should be documented. An acceptance test report should be produced describing the execution of the plan and summarizing the results. This report should contain a statement that the plan was successfully executed, and the system is ready for operation. The acceptance test report should demonstrate that the system operates correctly and is identical to the system that was validated during the validation phase. The report should summarize the test results after all problems have been satisfactorily resolved. The report should demonstrate that acceptance testing was executed according to the acceptance test procedure.

c. *Software Configuration Management Activities*

The software development plan describes the documents that will be created and placed under configuration management control. The configuration management plan describes the implementation tasks that are to be carried out by the applicant/licensee. The acceptance criterion for software CM implementation is that the tasks in that plan have been carried out in their entirety. Documentation should exist that shows that the configuration management tasks for that activity group have been successfully accomplished. In particular, the documentation should show that configuration items have been appropriately identified; that configuration baselines have been established for the activity group; that an adequate change control process has been used for changes to the product baseline; and that appropriate configuration audits have been held for the configuration items created or modified for the activity group.

Each configuration item should be labeled unambiguously so that a basis can be established for the control and reference of the configuration items defined in the software CM plan. Configuration baselines should be established for each life cycle activity group, to define the basis for further development, allow control of configuration items, and permit traceability between configuration items. The baseline should be established before the set of activities can be considered complete. Once a baseline is established, it should be protected from change. Change control activities should be followed whenever a derivative baseline is developed from an established baseline. A baseline should be traceable to the baseline from which it was established, and to the design outputs it identified or to the activity with which it is associated.

Configuration control actions should be used to control and document changes to configuration baselines. A configuration control board (CCB) should exist with the authority to authorize all changes to baselines. Problem reports should be prepared to describe anomalous and inconsistent software and documentation. Problem reports that require corrective action should invoke the change control activity. Change control should preserve the integrity of configuration items and baselines by providing protection against their change. Any change to a configuration item should cause a change to its configuration identification. This can be done via a version number or attached change date. Changes to baselines and to configuration items under change control should be recorded, approved and tracked. If the change is due to a problem report, traceability should exist between the problem report and the change. Software changes should be traced to their point of origin, and the software processes affected by the change should be repeated from the point of change to the point of discovery. Proposed changes should be reviewed by the CCB for their impact on system safety.

Status accounting should take place for each set of life cycle activities prior to the completion of those activities. The status accounting should document configuration item identifications, baselines, problem report status, change history and release status.

The configuration management organization should audit life cycle activities to confirm that configuration management procedures were carried out in the life cycle process implementation.

3.3. Acceptance Criteria for Software Life Cycle Process Design Outputs

This section describes the criteria to be used to determine whether the software has each of the characteristics important to safety system software. Criteria are organized first by life cycle activity group, then by design output, and then by characteristic.

Formal or semiformal methods are available for use in preparing some of the design outputs described in this section. Section C.3 of Appendix 7.0-A describes the benefits of using such methods and the precautions that should be observed when reviewing design outputs prepared with such methods.

Acceptance criteria are divided into two sets: functional characteristics and process characteristics, as shown in the following table. Not all characteristics occur for every design output.

Functional Characteristics	**Process Characteristics**
Accuracy	Completeness
Functionality	Consistency
Reliability	Correctness
Robustness	Style
Safety	Traceability
Security	Unambiguity
Timing	Verifiability

a. Requirements Activities — Software Requirements Specification

An SRS that exhibits the functional and the software development process characteristics listed below should be produced. Reg. Guide 1.172, "Software Requirements Specifications for Digital Computer Software Used in Safety Systems of Nuclear Power Plants," which endorses IEEE Std 830, "IEEE Recommended Practice for Software Requirements Specifications," describes an acceptable approach for describing software requirements.

Functional Characteristics

For each of the functional characteristics, the requirements imposed by the system requirements and system design on the software for that characteristic should be contained in the SRS. Functional characteristics addressed by the SRS include accuracy, functionality, reliability, robustness, safety, security, and timing.

Accuracy requirements should be provided for each input and each output variable. Accuracy requirements should be stated numerically, and appropriate physical units and error bounds should be supplied. Accuracy requirements should include a description of data type and data size for each input and output variable.

Functionality requires that the operations that must be performed for each mode of operation be completely specified. Functions should be specified in terms of inputs to the function, transformations to be carried out by the function, and outputs generated by the function.

Reliability requires that all requirements for fault tolerance and failure modes be fully specified for each operating mode. Software requirements for handling both hardware and software failures should be provided, including requirements for analysis of and recovery from computer system failures. Requirements for on-line in-service testing and diagnostics should be provided.

Robustness requires that the behavior of the software in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior be fully specified. Of particular concern is the behavior of the software in the presence of unexpectedly high or low rates of message traffic.

Safety requires that the software functions, operating procedures, input, and output be classified according to their importance to safety. Requirements important to safety should be identified as such in the SRS. The identification of safety items should include safety analysis report requirements, as well as abnormal conditions and events as described in Reg. Guide 1.152.

Security requires that security threats to the computer system be identified and classified according to severity and likelihood. Actions required of the software to detect, prevent, or mitigate such security threats should be specified, including access control restrictions.

Timing requires that functions that must operate within specific timing constraints be identified, and that timing criteria be specified for each. Timing criteria should be provided for each mode of operation. Timing requirements should distinguish between goals and requirements. Timing requirements should be stated in such a way that the time delay between stimulus and response for safety actions is deterministic under normal and anticipated failure conditions. BTP HICB-21 provides additional guidance on real-time performance.

Software Development Process Characteristics

Software development process characteristics exhibited by the SRS should include completeness, consistency, correctness, style, traceability, unambiguity and verifiability.

Completeness requires that all actions required of the computer system be fully described for all operating modes and all possible values of input variables (for example, the complete span of instrument inputs or clock/calendar time)[*]. The SRS should describe any actions that the software is prohibited from executing. The operational environment within which the software will operate should be described. All variables in the physical environment that the software must monitor and control shall be fully specified. Functional requirements should describe (1) how each function is initiated; (2) the input and output variables required of the function; (3) the task sequences, actions, and events required to carry out the function; and (4) the termination conditions and system status at the conclusion of the function. User interfaces should be fully described for each category of user.

Consistency requires that the contents of the SRS be consistent with the safety system requirements, the safety system design, and documented descriptions and known properties of the operational environment within which the safety system software will operate. Individual requirements should not contradict other requirements. Timing requirements should be consistent with thermohydraulic analyses performed in the system safety analysis. Uniform and consistent terminology, notation, and definitions should be used throughout the SRS.

Correctness requires that the description of actions required of the computer system be free from faults and that no other requirements be stated. The operational environment within which the software will operate should be accurately described. All variables in the physical environment that the software must monitor and control should be properly specified. Functional requirements should accurately describe (1) how each function is initiated; (2) the input and output variables required of the function; (3) the task

[*]Implementation of safety functions should not rely upon a date (calendar time). Actions depending upon calendar time should account for concerns such as those identified with the year 2000 (two-digit 00).

sequences, actions and events required to carry out the function; and (4) the termination conditions and system status at the conclusion of the function.

Style requires that the contents of the SRS be understandable. The SRS should differentiate between requirements placed on the software and other supplementary information, such as design constraints, hardware platforms, and coding standards. A precise definition of each technical term should exist, either in the SRS or in a separate dictionary or glossary. Each requirement should be uniquely and completely defined in a single location in the SRS.

Traceability requires that a two-way trace exist between each requirement in the SRS, and the safety system requirements and design. There should be a two-way trace between each requirement in the SRS and the software design, as well as a forward trace from each requirement in the SRS to the specific inspections, analyses, or tests used to confirm that the requirement has been met.

Unambiguity requires that each requirement, and all requirements taken together, have one and only one interpretation.

Verifiability requires that it be possible to construct a specific analysis, review, or test to determine whether each requirement has been met.

b. Design Activities — Software Architecture Description

A SAD should be produced. The SAD should include all of the functional and software development process characteristics listed below.

Functional Characteristics

For each of the functional characteristics, the requirements imposed on the software for that characteristic should be satisfied by the software architecture. A review of the software architecture requires a concurrent review of the hardware architecture. Functional characteristics addressed by the SAD should include reliability, safety, security, and timing.

Reliability requires that the combined hardware and software architecture be such that individual software element failure will not compromise safety. The software architecture should identify actions to be taken in the event of error detection. The hardware and software architecture should be reviewed to verify that the propagation of errors is controlled via a well-structured modular design.

Safety requires that the software architecture introduce no new hazards into the safety system. The safety functions should be separated from normal operating and overhead functions, with well-defined and strictly controlled interfaces between them. Any online maintenance features should be included. The hardware and software architecture should be reviewed to verify that there is no violation of other criteria such as single failure, channel separation, and separation between Class 1E and non-1E systems. The review should verify that no new hazards are introduced into the safety system as a result of the architecture configuration.

Security requires that the architecture correctly handle identified security threats, and introduce no new security threats.

Timing requires that the architectural design describe all timing limitations, the strategy for handling each, the required margins, and the method of measuring those margins. A timing specification should exist for each architectural element, in terms of minimum and maximum times for execution. Scheduling mechanisms and interprocess communication methods should be described. The architecture should be such that operations are performed in the correct sequence. BTP HICB-21 provides additional guidance on real-time performance. SRP Section 7.9 provides additional guidance on digital data communications systems.

Software Development Process Characteristics

The software development process characteristics that the SAD must exhibit include completeness, consistency, style, traceability, and verifiability.

Completeness requires that all the software requirements be satisfied in the architecture. The SAD should address all operating modes specified in the SRS, including initialization, operational, shut-down, maintenance, and test modes.

Consistency requires that each software architectural element be compatible with the SRS, the hardware architecture, documented descriptions and known properties of the operational and hardware environment, and other software elements. Timing specifications of each software element should be consistent with the specifications of the other elements with which it interacts and with the expected performance of the system as a whole. Uniform and consistent terminology, notation, and definitions should be used.

Style requires that the contents of the SAD be understandable. The architecture description should conform to the developer's style guide. The architecture specification should contain the rationale for architectural decisions.

Traceability requires that a two-way trace exist between the requirements in the SRS and the elements in the architecture. A two-way trace should exist between the architectural elements and the detailed design elements. There should be a forward trace from each architectural element to the specific inspections, analyses, or tests that will be used to confirm that the element has been correctly designed.

Verifiability requires that it be possible to construct specific analyses, reviews, and tests to verify that the architecture satisfies the software requirements.

c. Design Activities — Software Design Specification

An SDS should be produced. The SDS should include all of the functional and software development process characteristics listed below.

Functional Characteristics

For each of the functional characteristics, the requirements imposed on the software for that characteristic should be satisfied by the software design. Product functional characteristics addressed by the SDS should include accuracy, reliability, robustness, safety, security, and timing.

Accuracy requires that all calculations be specified in such a way that the accuracy requirements for the calculations will be satisfied. In particular, floating point arithmetic should be avoided; if that is not possible, special care must be taken to maintain the accuracy of the calculations. The design should specify the method for determining that the values of input variables are within the proper range, the method by which the software will detect that the values of input variables are not within their proper range, and the actions to be taken in the latter case. All calculations should be analyzed for convergence, round-off error, precision, and accuracy as appropriate.

Reliability requires that the detailed software design be such that single failures of individual elements will not cause safety system failure.

Robustness requires that the design be such that the software will operate correctly in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior. In particular, the software should not fail, and should not provide incorrect outputs, in the presence of these conditions. Attention should be paid to those values of input variables that are physically possible to the device, even if logically impossible in the application (to account for sensor errors, communication line noise, and similar concerns).

Safety requires that the detailed design introduce no new safety hazards into the safety system.

Security requires that unauthorized changes be prevented, detected, or mitigated as appropriate.

Timing requires that the time delay between stimulus and response be deterministic. BTP HICB-21 provides additional guidance on real-time performance.

Software Development Process Characteristics

The SDS should exhibit each of the following software development process characteristics: completeness, consistency, correctness, style, traceability, and verifiability.

Completeness requires that the detailed design specify the actions of each software unit for the entire domain of each input variable (for example, the complete span of instrument inputs or clock/calendar time). The design should be sufficiently complete to permit implementation to take place. Actions should be specified for all situations anticipated in the SRS. Equipment, human, hardware, and software interfaces should be correctly and fully specified. Equations, algorithms, and control logic should be correctly and fully specified.

Consistency requires that the detailed design be consistent with the architectural design, and that the detailed design elements be mutually consistent. Design elements should be consistent with documented descriptions and known properties of the operational environment within which the software will execute. Input and output specifications specified in the software design should be consistent with interface requirements imposed by the hardware or predeveloped software products. Timing specifications of each detailed design element should be consistent with the timing specifications of the architectural element of which it is a part. Models, algorithms, and numerical techniques specified in the software design should agree with standard references where such are applicable. A uniform and consistent terminology, notation, and definitions should be used. Models, algorithms, and numerical techniques specified in the software design should be mathematically mutually compatible.

Correctness requires that all equations, algorithms, and control logic be evaluated for potential errors. All equations and algorithms should be defined to a sufficient level of detail to permit coding. Data structure design should ensure that the code elements will correctly initialize data, correctly access stored data, and correctly scale and dimension data. The detailed design should ensure that no data item can be used before it is initialized, can have its value changed in an unanticipated manner, or can have its value changed by an unanticipated design element. The detailed design should ensure that no data item can be changed in an unanticipated manner.

Style requires that the detailed design documents description should conform to the developer’s style guide. Each element of the detailed design should be specified. The detailed design documentation should contain the rationale for design decisions. Programming language standards should be identified. The detailed design documentation should identify those language features which will not be used without justification.

Traceability requires that a two-way trace exist between the elements of the detailed design and the elements in the architecture. A two-way trace should exist between the detailed design elements and the code elements. There should be a forward trace from each detailed design element to the specific inspections, analyses, or tests that will be used to confirm that the element has been correctly designed.

Verifiability requires that it be possible to construct specific analyses, reviews, and tests to verify that the design satisfies the software architecture.

d. Implementation Activities — Code Listings

A software implementation (code) should be produced. The code should include all of the functional and software development process characteristics listed below.

Functional Characteristics

For each of the functional characteristics, the requirements imposed on the software for that characteristic should be satisfied by the code. Functional characteristics addressed by the code documents should include accuracy, robustness, safety, and timing.

Accuracy requires that the actual source code be written so that the accuracy requirements and accuracy design specifications are met. In particular, special care should be taken for floating point arithmetic, round-off errors, and the retention of precision during numerical operations. If mathematical subroutine libraries are used, the accuracy characteristics of the subroutines should be known and documented, and shown to meet the accuracy requirements and accuracy design specifications.

Robustness requires that the system be coded in such a way that corrupted data will not cause the safety system to fail. Data corruption should be avoided. All input data should be checked to ensure that the correct data is being read and that the data is in the correct format. All messages should be checked to ensure that the correct message is being read and that the message contents are in the correct format. Appropriate corrective actions should take place if any of these criteria are violated.

Safety requires that the code introduce no new hazards into the safety system.

Security requires that the code introduce no new security threats into the safety system software.

Timing requires that the execution time be deterministic. BTP HICB-21 provides additional guidance on real-time performance.

Software Development Process Characteristics

Software development process characteristics exhibited by the code documents should include completeness, consistency, correctness, style, traceability, and verifiability.

Completeness requires that the code meet all the specifications of the design and all implementation constraints. The software implementation should be compatible with the hardware environment.

Consistency requires that all variable names, types, locations, and array sizes be defined consistently throughout the software units. The code should use mathematical equations which correspond to the mathematical models, algorithms, and numerical techniques described in or derived from the SDS. All parameters passed between software units should be consistent with respect to number, type, structure, physical units, and direction. Minimum and maximum execution times should be consistent with expected overall performance.

Correctness requires that the code be correctly implemented.

Style requires that the programming style constraints specified in the design documents be followed. NUREG/CR-6463, "Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems," provides guidance on coding practices to be avoided. In particular, data structures should be protected so that they cannot be changed simultaneously. Arrays should have a fixed, predefined length. Global variables and dynamic memory allocation should not be used.

Traceability requires that a two-way trace exist between the elements of the detailed design and the elements in code. A two-way trace should exist between the code elements and the specific software subsystem or system which contains that element during factory build and test. There should be a forward trace from each detailed design element to the specific inspections, analyses, or tests that will be used to confirm that the element has been correctly implemented.

Verifiability requires that it be possible to construct specific analyses, reviews, and tests to verify that the code correctly implements the detailed design.

e. Integration Activities — System Build Documents

One or more system build documents should be produced. The build documents should include all of the functional and software development process characteristics listed below.

Functional Characteristics

For each of the functional characteristics, the requirements imposed on the build documents for that characteristic should be satisfied. Functional characteristics addressed by the build documents should include robustness, safety, and security.

Robustness requires that the software build documents specify methods to detect incorrectly built software releases. The software build documents should identify all errors and anomalies discovered during software build activities.

Safety requires that the software build activity introduce no new hazards into the safety system.

Security requires that the software build activity introduce no new security threats into the safety system software.

Software Development Process Characteristics

The system build documents must exhibit each of the following software development process characteristics: completeness, consistency, correctness, style, traceability, and verifiability.

Completeness requires that all build procedures be fully specified. The software build documents should include all required software units, including code and data, that are part of the build.

Consistency requires that the software build documents be consistent with the software specifications, as described in the SRS, software design description, and software code. A consistent and uniform set of terminology, notation, and definitions should be used throughout the software build document.

Correctness requires that the software build documents identify the correct versions of all required software elements and all required software documents. It should be verified that the correct elements have actually been used in the build, including proper units from software libraries.

Style requires that the software build documents conform to applicable standards imposed by the developer. A precise definition of each technical term used in the build documents should be included in the document, or in a separate dictionary or glossary.

Traceability requires that it be possible to trace each element of the integrated builds (software subsystem or software system) backward to the code elements contained in the build. It should be possible to trace each element of the integrated build forward to the software field installation.

Verifiability requires that it be possible to analyze, review, or test each integrated software build for the product functional requirements. The system build documents should specify methods to detect incorrectly built software releases. The build documents should identify all errors and anomalies discovered during software build activities.

f. Installation Activities — Installation Configuration Tables

Installation configuration tables should be produced. They should include all of the functional characteristics listed below to ensure that the software will be correctly configured in the operating safety system. The software development process characteristics listed below should be exhibited by the installation configuration tables themselves.

Functional Characteristics

For each of the functional characteristics, the requirements imposed on the configuration tables for that characteristic should be satisfied. Functional characteristics addressed by the configuration tables should include functionality, safety, and security.

Functionality requires that the installation tables configure the installed system to have the functionality that is required for the plant.

Safety requires that the installation tables introduce no new hazards into the safety system.

Security requires that the installation tables introduce no new security threats into the installed system, and that the installation tables be protected from unauthorized change.

Software Development Process Characteristics

The configuration tables must exhibit each of the specified software development process characteristics: completeness, consistency, correctness, traceability, and verifiability.

Completeness requires that the software configuration tables include all information necessary for the correct operation of the system.

Consistency requires that the installation configuration tables be consistent with the software specifications, as described in the SRS, software design description, software code, and software build documents.

Correctness requires that the software configuration tables contain all plant-specific data.

Traceability requires that it be possible to trace each installed program element backward to the integrated software elements that created that installed program element.

Verifiability requires that it be possible to analyze, review, or test each installed software system on initial software installation, all subsequent installations, and periodically during operation.

g. Installation Activities — Operations Manuals

One or more software operations manuals should be produced. They may be incorporated into a system operations manual. Because operations manuals do not impose requirements on the software itself, the functional characteristics described in other sections of this BTP are not directly relevant. However, the software development process characteristics listed below should be exhibited by software operations manuals.

Completeness requires that all actions available to the system operator be fully described for all operating modes, including error recovery and backup. Operator actions should be specified in terms of inputs supplied by the operator or equipment, actions initiated by the operation, and responses to the operator. The purpose and operation of each function should be described, including interfaces with other functions. The operations manual should describe the operational environment within which the software will operate, including precautions and limitations that must be observed during operations to avoid exposing personnel or the plant to hazards or security vulnerabilities. All variables in the physical environment that the software must monitor and control should be fully described. User interfaces should be fully described for each category of user.

Consistency requires that the operations manual be consistent with the system operations, safety system requirements, the safety system design, the SRSs, the SDS, and documented descriptions and known properties of the operational environment within which the safety system will operate. Individual user instructions should not contradict other instructions. Uniform and consistent terminology, notation, and definitions should be used throughout the operations manuals.

Style requires that the operations manual be understandable by the users of the manual. A precise definition of each technical term should exist, either in the operations manual or in a separate dictionary or glossary. The operations manual may be organized in the style of a reference manual, with the assumption that its users are well trained.

Traceability requires that a forward trace exist between the SRS, the operations plan, and the operations manual, which shows how each requirement is to be carried out by the operators, or carried out automatically by the safety system without operator action, and how the results of each requirement are displayed to the operators. A forward trace should also exist from all error messages generated by the code to a description of the error messages in the operations manual.

*Unambiguit*y requires that instructions to users have only one interpretation by the users.

h. Installation Activities — Maintenance Manuals

One or more software maintenance manuals should be produced. They may be incorporated into a system maintenance manual. Because maintenance manuals do not impose requirements on the software itself, the functional characteristics described in other sections of this BTP are not directly relevant. However, the software development process characteristics listed below should be possessed by software maintenance manuals.

Completeness requires that maintenance procedures be fully defined. This should include identification of precautions and limitations that must be observed during maintenance to avoid exposing personnel or the plant to hazards or security vulnerabilities. Trouble reports should be collected from field installations and analyzed to determine if changes to the software are required. Configuration management procedures should be described in or referenced by the maintenance manual. Procedures should exist to (1) verify that changes have been carried out correctly and that no faults have been introduced in the software by the changes, and (2) ensure that software is correctly returned to service. Field upgrade procedures should be described.

Style requires that the maintenance manual be understandable by the users. A precise definition of each technical term should exist, either in the maintenance manual or in a separate dictionary or glossary. The maintenance manual may be organized in the style of a reference manual, with the assumption that its users are well trained.

Traceability requires that a forward trace exist between the maintenance plan and the maintenance manual, which shows how each requirement is carried out by the maintenance organization.

i. Installation Activities — Training Manuals

One or more software training manuals should be produced. They may be incorporated into a system training manual. Because training manuals do not impose requirements on the software itself, the functional characteristics described in other sections of this BTP are not directly relevant. However, the software development process characteristics listed below should be exhibited by software training manuals.

Completeness requires that all actions available to the operator be fully described for all operating modes, including error recovery. Operator actions should be specified in terms of inputs supplied by users and equipment, actions initiated by the operation, and responses to the user. The training manual should describe the operational environment within which the software will operate, including precautions and limitations that must be observed during operations to avoid exposing personnel or the plant to hazards. All variables in the physical environment that the software must monitor and control should be fully described. User interfaces should be fully described for each category of user.

Consistency requires that the training manual be consistent with the safety system requirements, the safety system design, the SRSs, the SDS, and documented descriptions and known properties of the operational environment within which the safety system will operate. Individual user instructions should not contradict other instructions. Uniform and consistent terminology, notation, and definitions should be used throughout the training manuals.

Style requires that the training manual be understandable by the users. A precise definition of each technical term should exist, either in the training manual or in a separate dictionary or glossary. The operations manual may be organized in the style of a tutorial guide, with the assumption that the users also have access to the operations manual.

Traceability requires that a forward trace exist between the SRS, the training plan, and the training manual, which shows how each requirement is to be carried out by the users, or carried out automatically by the safety system without user action, and how the results of each requirement are displayed to the users.

4. Review Procedures

Reviews are carried out by a combination of inspection and analysis of documents. The adequacy of the computer development process should be reviewed to confirm that software life cycle plans incorporate appropriate commitments, as described in Section 3.1 above. New software, or an unproven development team, will require greater emphasis on the adequacy of the planning phase. A sample of V&V, safety analysis, and configuration management documentation for various life cycle activity groups should be audited to confirm that the developer’s life cycle activities have been properly implemented. Section 3.2 above presents specific criteria from which the inspection activities for each specific life cycle activity may be derived. A sample of software design outputs should be reviewed to confirm that they address the functional requirements allocated to the software, and that the expected software development process characteristics are evident in the design outputs. Section 3.3 above describes functional characteristics and software development process characteristics from which the inspection activities for each specific design output may be derived. SRP Appendix 7.0-A contains additional detail on the software review process and the relationship between software reviews and system reviews.

C. References

ANSI/IEEE Std 1008-1987. "IEEE Standard for Software Unit Testing."

ANSI/IEEE Std 1012-1986. "IEEE Standard for Software Verification and Validation Plans."

ANSI/IEEE Std 1058.1-1987. "IEEE Standard for Software Project Management Plans."

ANSI/IEEE Std 279-1971. "Criteria for Protection Systems for Nuclear Power Generating Stations."

ANSI/IEEE Std 829-1983. "IEEE Standard for Software Test Documentation."

ASME Std NQA-1-1994. "Quality Assurance Requirements for Nuclear Facility Applications."

EPRI Topical Report TR-106439. "Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications." Electric Power Research Institute, October 1996.

IEEE Std 1028-1988. "IEEE Standard for Software Reviews and Audits."

IEEE Std 1042-1987. "IEEE Guide to Software Configuration Management."

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

IEEE Std 1219-1992. "IEEE Standard for Software Maintenance."

IEEE Std 1228-1994. "IEEE Standard for Software Safety Plans."

IEEE Std 603-1991. "IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations."

IEEE Std 7-4.3.2-1993. "IEEE Standard for Digital Computers in Safety Systems of Nuclear Power Generating Stations."

IEEE Std 730.1-1989. "IEEE Standard for Quality Assurance Plans."

IEEE Std 828-1990. "IEEE Standard for Software Configuration Management Plans."

IEEE Std 830-1993. "IEEE Recommended Practice for Software Requirements Specifications."

NUREG/CR-6421. "A Proposed Acceptance Process for Commercial Off-the-Shelf (COTS) Software in Reactor Applications." March 1996.

NUREG/CR-6101. "Software Reliability and Safety in Nuclear Reactor Protection Systems." 1993.

NUREG/CR-6463. "Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems." June 1996.

Regulatory Guide 1.152. "Criteria for Digital Computers in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, January 1996.

Regulatory Guide 1.168. "Verification, Validation, Reviews and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

Regulatory Guide 1.169. "Configuration Management Plans for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

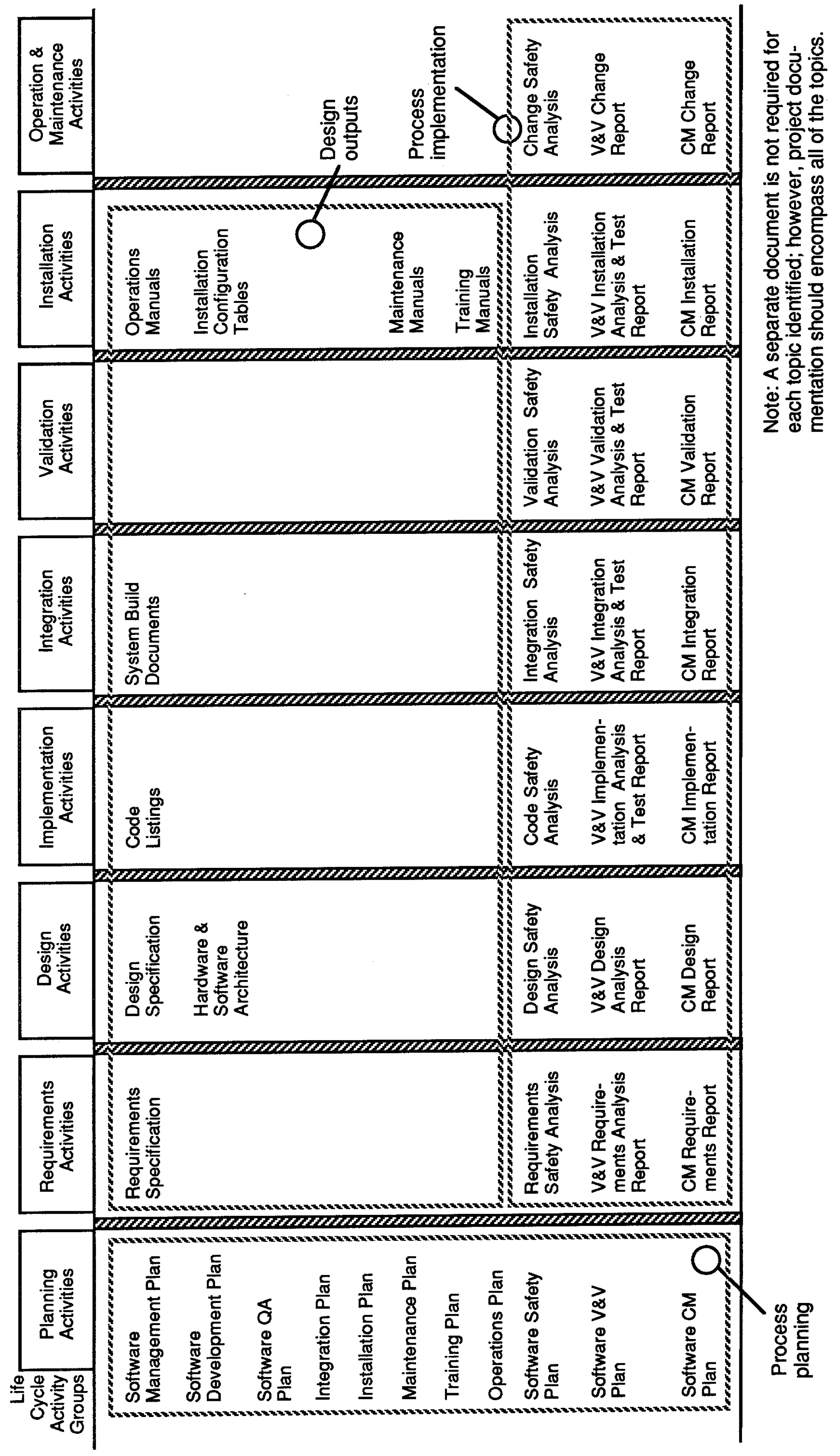
Regulatory Guide 1.170. "Software Test Documentation for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

Regulatory Guide 1.171. "Software Unit Testing for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

Regulatory Guide 1.172. "Software Requirements Specifications for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

Regulatory Guide 1.173. "Developing Software Life Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1997.

Safety Evaluation by the Office of Nuclear Reactor Regulation. "EPRI Topical Report TR-106439." May 1997.



Note: A separate document is not required for each topic identified; however, project documentation should encompass all of the topics.

Figure 7-A-1. Flow of Documents Through the Software Life Cycle