

Enclosure
Attachment 13
PG&E Letter DCL-11-104

**DCPP Procedure CF2.ID9, Revision 2,
“Software Quality Assurance for Software Development”
(LAR Reference 51)**

TITLE: Software Quality Assurance for Software Development

INFO ONLY
EFFECTIVE DATE

PROCEDURE CLASSIFICATION: QUALITY RELATED

TABLE OF CONTENTS

<u>SECTION</u>	<u>PAGE</u>
SCOPE	2
DISCUSSION	2
DEFINITIONS	3
RESPONSIBILITIES	4
INSTRUCTIONS	5
Design Phase	5
Conceptual Design	5
Functional Requirements Specification (FRS)	6
Implementation Phase	7
Software Requirements Specification (SRS)	7
Software Design Specification (SDS)	7
Software Design Description (SDD)	9
Application Development	10
Software Installation Plan (SIP)	11
Software Verification and Validation Plan (SVVP)	11
Software Verification and Validation Report (SVVR)	12
Installation	12
Installation Testing	12
Operation and Maintenance Phase	13
Software Configuration Management Plan (SCMP)	13
Operation and Maintenance Phase	13
Operation and Maintenance Phase	13
RECORDS	14
APPENDICES	14
ATTACHMENTS	14
REFERENCES	14

TITLE: Software Quality Assurance for Software Development

1. SCOPE

- 1.1 The purpose of this procedure is to outline the process utilized for the in-house development of new power plant related software applications.
- 1.2 This procedure applies to the development of all new software applications (quality and non-quality related) that affect power plant operation. Applications with functions that affect power plant operation include (but are not limited to):
 - Plant Process Monitoring (scan, log, and alarm)
 - Plant Process Control
 - Other applications related to power plant performance
- 1.3 This procedure is not intended for development of new power or changes to existing plant applications that have an approved SQA Plan.

2. DISCUSSION

- 2.1 For the purpose of this procedure, software development will be divided into the design, implementation, and maintenance phases. Section 5 discusses each phase in depth. Figure 1 depicts the process diagram.
- 2.2 Guidelines in this procedure are considered good practices when developing new applications. Although it is not intended to govern business applications or information technology applications (e.g., desktop workstations, calculators, analysis programs, and other engineering tools) it can be used as a guideline in developing such applications.

TITLE: Software Quality Assurance for Software Development

- 2.3 Application development using the process described by this procedure provides for dedication of any Commercial Off-the-Shelf (COTS) software incorporated into the application.

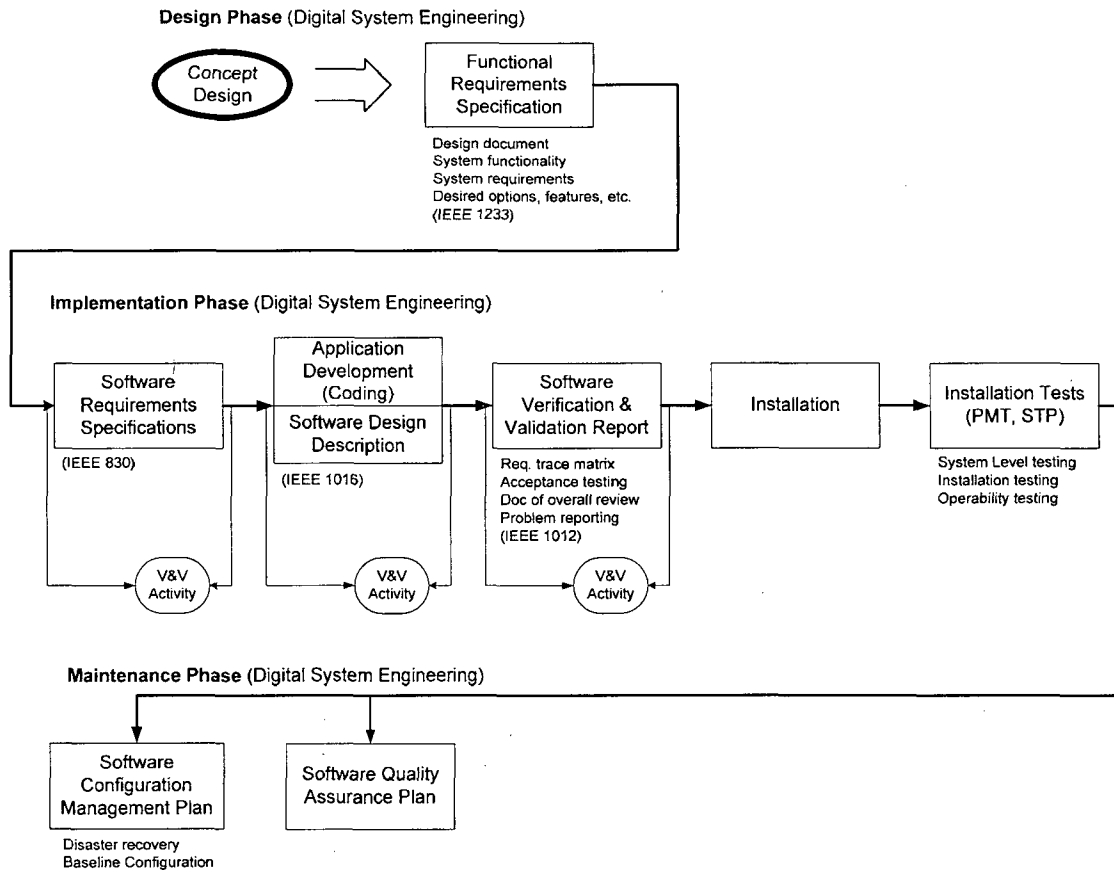


Figure 1: The Software Life Cycle

3. DEFINITIONS

- 3.1 Baseline: A specification or system that has been formally reviewed and agreed upon; and thereafter serves as the basis for further development and can be changed via a formal change process.
- 3.2 Requirement: A statement that identifies an operational, functional, or design characteristic or constraint. A requirement must be unambiguous, testable or measurable, and necessary for acceptability.
- 3.3 Validation: The process of evaluating a system or component during or at the end of the development process to determine whether a system or component satisfies the requirements.
- 3.4 Verification: The process of evaluating a system or component to determine that the system of a given development phase satisfies the conditions imposed at the start of that phase.

TITLE: Software Quality Assurance for Software Development

4. RESPONSIBILITIES

- 4.1 Design engineering is responsible for:
 - 4.1.1 Overall responsibility for the Design Phase.
 - 4.1.2 Development of the Functional Requirements Specification.
- 4.2 Project engineering is responsible for:
 - 4.2.1 Overall responsibility for the Implementation Phase.
 - 4.2.2 Development of Software Requirements Specification, Software Design Specification, Software Design Description, Software Installation Plan, Software Verification and Validation Plan, Software Verification and Validation Report, Software Configuration Management Plan and Software Quality Assurance Plan.
- 4.3 Configuration coordinator and test group are responsible for determining what testing is required to ensure system function prior to being placed in service.
- 4.4 Digital systems engineering has overall responsibility for the Operations and Maintenance Phase.

TITLE: Software Quality Assurance for Software Development

5. INSTRUCTIONS

5.1 Design Phase

This includes coordination during conceptual design, design change activities, and functional requirements specifications.

5.1.1 Conceptual Design

The purpose of the conceptual design process is to solicit all stakeholders and extract a complete set of requirements ensuring that none are missed.

Because of the dynamics involved in identifying requirements, it is important to focus the process toward developing a functional requirements specification.

- a. Throughout the conceptual design process ensure the following:
 1. The process is goal directed and aimed at the production of a set of requirements.
 2. The system boundaries are defined.
 3. Requirements are solicited from all stakeholders, fairly evaluated, and documented.
 4. Requirements are specified as capabilities, and qualifying conditions and bounding constraints are identified distinctly from capabilities.
 5. Requirements are validated, or purged, if invalid, from the requirements set.
 6. Consideration is given to consistency when many individuals may be contributing to the development of the requirements set.
 7. The developing requirements set is understood, at an appropriate level of detail, by all individuals participating in the process.

b. Verification and Validation

There are no verification requirements for this phase.

TITLE: Software Quality Assurance for Software Development

5.1.2 Functional Requirements Specification (FRS)

Refer to Attachment 8.1 for guidelines.

Refer to Attachment 8.2 for sample outline.

The purpose of the FRS is to provide a "black box" description of what the system should do, in terms of the system's interactions or interfaces with its external environment. All inputs, outputs, and required relationships between inputs and outputs as well as any human/machine interfaces should be completely described.

The FRS is a document that communicates the system requirements from the designing organization to the implementing organization, acting as a bridge between the two.

The FRS is a formally documented baseline which contains system functional and performance requirements, including any design or licensing basis information. The FRS should not contain process requirements (detail relative to contract or workflow).

a. Verification and Validation

Independent review of the functional requirements specification document meets the verification requirements for this phase.

TITLE: Software Quality Assurance for Software Development

5.2 Implementation Phase

This includes documentation of: software requirements, software design description, application development, detailed software design, software installation plan, verification and validation plan, acceptance tests, problem identification, verification and validation report.

5.2.1 Software Requirements Specification (SRS)

Refer to Attachment 8.1 for guidelines.

Refer to Attachment 8.3 for sample outline.

Consider the part that the SRS plays in the total project plan. The software may contain essentially all the functionality of the project or it may be part of a larger system. In the latter case, there will be a FRS that will state the interfaces between the system and its software portion, and will place external performance and functionality requirements upon the software portion. The SRS must agree with and should expand upon these functional requirements.

The SRS should correctly define all the software requirements. A software requirement may exist because of the nature of the task to be solved or because of a special characteristic of the project. The SRS should not describe any design or implementation details or impose additional constraints on the software, the SRS should be careful not to go beyond these bounds.

a. Verification and Validation

Independent review of the software requirements specification document meets the verification requirements for this phase.

5.2.2 Software Design Specification (SDS)

For complex systems, the SDS should be produced as a separate document. Otherwise, the content may be included as part of the SDD. In either case, the following information shall be documented:

a. Design Architecture

The design architecture describes the hardware and software elements, and the mapping of the software into the hardware. The hardware architecture shows the various hardware items – computers, file systems, I/O devices (such as sensors, actuators and terminals) and communication networks. This provides the view of:

1. Where computations physically take place
2. Where information is physically stored and displayed
3. How information is physically routed from one device to another

TITLE: Software Quality Assurance for Software Development

b. Software Architecture

The software architecture shows the various software processes, databases, files, messages and screen designs. This provides the logical view of:

1. Where computations logically take place
2. Where information is logically stored and displayed
3. How information moves logically from one process, data store or input device to another process, data store or output device

c. Mapping

Mapping shows the connections between the software and the hardware. Each process runs on a computer, each data store resides on one or more storage media, each screen display is shown on a terminal, and information moves over physical communication lines in order to carry out logical communication between processes and I/O devices.

d. Independent Review

1. The reviewer of the software architecture should be able to understand how the software works, the flow of data, and the deterministic nature of the software. The architecture should be sufficiently detailed to allow the reviewer to understand the operation of the software.
2. The software architecture should satisfy the software requirements for each identified functional characteristic. The review of each function should include an evaluation of reliability, safety, security and timing.
3. The SRS requirements should be completely traceable to the elements in the software architecture. The process characteristics should exhibit completeness, consistency, conformity of style and verifiability.

TITLE: Software Quality Assurance for Software Development

5.2.3 Software Design Description (SDD)

The purpose of the SDD is to provide an exact description of how the software requirements are to be implemented in design elements. It is a blueprint or model of the software system. The SDD should depict a hierarchical decomposition into layers of structurally and functionally distinct design elements. Design elements may be components and sub-components of the software system, software modules, databases, files, data structures or data stores, messages, programs or processes. Each design element is considered a configuration item and should be placed under software configuration management. Typical contents include functional elements, control logic, data structures, input/output formats, interface descriptions, and algorithms.

Consideration should be given to the software design methods when developing the SDD. The application may contain various combinations of design methods. The objective is to clearly document how the software is designed, rather than prescribe a method of documentation for a given design method.

a. Some examples of common design methods:

1. Function-oriented
2. Data-oriented
3. Real-time control-oriented
4. Object-oriented
5. Formal language-oriented

b. Verification and Validation

Independent review of the software design document and source code meets the verification requirements for this phase.

TITLE: Software Quality Assurance for Software Development

5.2.4 Application Development

The application is developed using the Software Design Description. Initially the SDD may be more of an outline of the software structure. As the application is further developed, or later modified, the SDD shall be revised to reflect the actual software design.

There are various methods and languages for developing software applications. It is not possible to provide specific development instructions. (Reference 9.21)

- a. Regardless of the developmental methods, the following four top level safety related attributes define the general quality of software.
 1. Reliability. The predictable and consistent performance of the software under conditions specified in the design basis. This is important to safety because it decreases the likelihood that faults causing unsuccessful operation will be introduced into the source code during implementation.
 2. Robustness. Robustness is the capability of the safety system software to operate in an acceptable manner under abnormal conditions or events. This is important to safety because it enhances the capability of the software to handle exception conditions, recover from internal failures, and prevent propagation of errors arising from unusual circumstances.
 3. Traceability. Traceability relates to the feasibility of reviewing and identifying the source code and library component origin and development processes, i.e., that the delivered code can be shown to be the product of a disciplined implementation process. Traceability also includes being able to associate source code with higher level design documents. This is important to safety because it facilitates verification and validation, and other aspects of software quality assurance.
 4. Maintainability. The means by which the source code reduces the likelihood that faults will be introduced during changes made after delivery. This is important to safety because it decreases the likelihood of unsuccessful operation resulting from faults during adaptive, corrective, or perfective software maintenance.
- b. When possible, each software module should contain a banner containing the following.
 1. Title. The name of the software application.
 2. Description. A brief description of the software application. This should include any inputs, outputs, dependencies, & etc.
 3. Author and Date. The author and date of the software application.
 4. Revision History. The revision number, or date, name of person, and a brief description of the change.

TITLE: Software Quality Assurance for Software Development

c. Verification and Validation

Independent review of the software design document and review of the source code meets the verification requirements for this phase.

5.2.5 Software Installation Plan (SIP)

A Software Installation Plan describes the tasks to be performed during installation, the required environment and other constraints, detailed instructions for the installer, and any additional steps that are required prior to the operation of the system.

a. Installation Environment

Describe the environment within which the software product is expected to perform. This can include the reactor itself, the reactor protection system, and the protection system instrumentation and computer hardware. This description should be limited to those items required for successful installation and operation.

b. Installation Package

Describe all of the materials that will be included in the installation package. This will include the software products, the media that contain them, and associated documents.

c. Installation Procedures

Describe completely the procedure for installing the software in the operational environment. This should be a step-by-step procedure, written for the customer. Anticipated error conditions should be described, with the appropriate recovery procedures.

5.2.6 Software Verification and Validation Plan (SVVP)

Refer to Attachment 8.4 for sample outline.

The purpose of the SVVP is to identify and describe the methods to be used to:

a. Verify that:

1. The requirements in the FRS have been reviewed and approved by the appropriate authority.
2. The requirements in the SRS have been reviewed and approved by the appropriate authority.
3. The design as described in the SDD is implemented in the code.

b. Validate that the code, when executed, complies with the requirements expressed in the FRS and SRS.

c. Verification and Validation

Independent review of the software verification and validation plan meets the verification requirements for this phase.

TITLE: Software Quality Assurance for Software Development

5.2.7 Software Verification and Validation Report (SVVR)

- a. The SVVR is the place where the requirements and design come together. This document contains the following:
 1. Requirements Traceability Matrix.
Generally a tabular presentation of all the requirements, such that various acceptance test and V&V activities can be linked. May not be appropriate for smaller applications with only a handful of requirements.
 2. Acceptance Test.
Generally a comprehensive test to demonstrate compliance with the requirements. May also include steps to demonstrate application capabilities or features. For larger applications, may be divided into section and organized as to functionality.
 3. Problem Report Log.
Prior to start of acceptance testing a problem reporting log shall be established. Refer to Attachment 8.5. This is used to document any anomalies (either in the test or application) as well as corrective actions and retest (if required).
 4. Summary of Reviews and/or Comments.
This should include final review of all project documentation. Documentation of any additional activities as specified in the SVVP.
- b. Verification and Validation
Independent review of the software verification and validation report meets the verification requirements for this phase.

5.2.8 Installation

Installation into the plant in accordance with the SIP and AD7.DC8, "Work Control." No additional steps are required by this procedure.

5.2.9 Installation Testing

- a. Post Modification Testing
Post modification testing in accordance with AD13.ID2, "Post Modification Testing." Generally this testing ensures proper installation and overall system functionality.
- b. Surveillance testing
Surveillance testing in accordance with AD13.DC1, "Control of the Surveillance Test Program." Generally this testing is used to determine Tech. Spec. Operability.

TITLE: Software Quality Assurance for Software Development

5.3 Operation and Maintenance Phase

5.3.1 Software Configuration Management Plan (SCMP)

A Software Configuration Management Plan shall be in place prior to declaring the system operational. The SCMP provides for maintaining the software lifecycle past the development stage into retirement.

Refer to Attachment 8.6 for sample outline.

- a. Configuration control practices are mandatory for every quality related structure, system and component to ensure that what is required to be in the plant (design and licensing bases), what is actually there (the physical plant), and what we say is there (plant procedures, drawings, documents, etc.) are all in alignment.
 - 1. All configuration changes must be auditable
 - 2. All configuration changes must be authorized
 - 3. All unauthorized configuration changes must be investigated, documented and resolved. No unauthorized change should remain in the environment.
- b. The SCMP summarizes how conformance with CF1, "Configuration Management" program requirements will be achieved. The SCMP provides the methods and tools to identify and control software throughout its use. Activities include:
 - 1. Identification and establishment of baselines
 - 2. Review, approval, and control of changes
 - 3. Tracking and reporting of such changes
 - 4. Audits and reviews of the evolving products
 - 5. Control of interfaces

5.3.2 Software Quality Assurance Plan (SQAP)

A Software Quality Assurance Plan shall be in place prior to declaring the system operational. This SQAP provides for maintaining the software lifecycle past the development stage into retirement. The SQAP should show how the overall responsibility for the software is retained, and how the quality of the software is determined to be sufficient. The SQA Plan is developed in accordance with this procedure.

5.3.3 Modifying approved SCMP and SQAP

Changes to approved SCMP and/or SQAP require the performance of a LBIE Screen, per TS3.ID2, and approval of the Owner/Supervisor of the SCMP/SQAP.

TITLE: Software Quality Assurance for Software Development

6. RECORDS

All documentation shall be maintained in accordance with AD10.ID1, "Storage and Control of Quality Records." The following documentation steps are required as a minimum. Depending upon the complexity and/or safety significance of the application, each of the documentation steps may be separate documents or sections of combined documents.

- Functional Requirements Specification (FRS)
- Software Requirements Specification (SRS)
- Software Design Specification (SDS)
- Software Design Description (SDD)
- Software Installation Plan (SIP)
- Software Verification and Validation Plan (SVVP)

NOTE: The SVVR contains the Requirements Traceability Matrix, Acceptance Tests, and Problem Report log.

- Software Verification and Validation Report (SVVR)
- Software Configuration Management Plan (SCMP)
- Software Quality Assurance Plan (SQAP)

7. APPENDICES

None

8. ATTACHMENTS

- 8.1 "Requirements Guidelines," 07/05/00
- 8.2 "Functional Requirements Specification Outline," 07/05/00
- 8.3 "Software Requirements Specification Outline," 07/05/00
- 8.4 "Software Verification and Validation Plan Outline," 07/05/00
- 8.5 Form 69-20322, "Problem Reporting and Corrective Action," 07/05/00
- 8.6 "Software Configuration Management Plan Outline," 06/01/10

9. REFERENCES

- 9.1 ANS 10.3-1995: Standard for Documentation of Computer Software
- 9.2 ANS 10.4-1987: Standard for Guidance for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry
- 9.3 ASME NQA-1-1997 (subpart 2.7): Quality Assurance Requirements for Nuclear Facility Applications - Quality Assurance Requirements for Computer Software for Nuclear Facility Applications
- 9.4 IEEE 7-4.3.2-1993: Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations
- 9.5 IEEE Std. 730-2002: IEEE Standard for Software Quality Assurance Plans

TITLE: Software Quality Assurance for Software Development

- 9.6 IEEE Std. 828-2005: IEEE Standard for Software Configuration Management Plans
- 9.7 IEEE 829-1983: Standard for Software Test Documentation
- 9.8 IEEE 830-1998: Recommended Practice for Software Requirements Specifications
- 9.9 IEEE 1008-1987: Standard for Software Unit Testing
- 9.10 IEEE 1012-1998: Standard for Software Verification and Validation
- 9.11 IEEE 1016-1987: Recommended Practice for Software Design Descriptions
- 9.12 IEEE 1016.1-1993: Guide to Software Design Descriptions
- 9.13 IEEE 1059-1993: Guide for Software Verification and Validation Plans
- 9.14 IEEE 1074-1995: Standard for Developing Software Life Cycle Processes
- 9.15 IEEE 1233-1998: Guide for Developing System Requirements Specifications
- 9.16 EPRI TR-103291: Handbook for Verification and Validation of Digital Systems
- 9.17 EPRI TR-106439: Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications
- 9.18 NUREG/CR-6241: A Proposed Acceptance Process for Commercial Off the Shelf (COTS) Software in Reactor Applications
- 9.19 NUREG/CR-6101: Software Reliability and Safety in Nuclear Reactor Protection Systems
- 9.20 NUREG/CR-6303: Method for Performing Diversity and Defense-in-Depth Analysis of Reactor Protection Systems
- 9.21 NUREG/CR-6463 Rev. 1: Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems
- 9.22 10 CFR, Part 50, Appendix B: Quality Assurance Criteria for Nuclear Power Plants and Fuel Reprocessing Plants
- 9.23 SRP Branch Technical Position (BTP) 7-14: Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems
- 9.24 Regulatory Guide 1.168: Verification, Validation, Reviews, and Audits for Digital Computer Software used in Safety Systems of Nuclear Power Plants, Revision 1
- 9.25 Regulatory Guide 1.170: Software Test Documentation for Digital Computer Software Used in Safety Systems of Nuclear Power Plants
- 9.26 Regulatory Guide 1.171: Software Unit Testing for Digital Computer Software used in Safety Systems of Nuclear Power Plants
- 9.27 Regulatory Guide 1.173: Developing Software Life Cycle Processes for Digital Computer Software used in Safety Systems of Nuclear Power Plants

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.1

TITLE: Requirements Guidelines

1. Common-Mode Failure Considerations

- a. Redundant trains of digital I&C systems may share databases (software) and process equipment (hardware). Therefore a hardware design error, software design error or software programming error may result in a common-mode or common-cause failures. Common-mode failure could defeat the redundancy achieved by the hardware architectural structure and could result in the loss of more than one echelon of defense-in-depth provided by the monitoring, control, reactor protection and engineering safety functions performed by the digital I&C systems. The two principal factors for defense against common-mode and common-cause failures are quality and diversity. Maintaining high quality will increase the reliability of both individual components and complete systems. Diversity in assigned functions (for both equipment and human activities), equipment, hardware and software can reduce the probability that a common-mode failure will propagate.
- b. NUREG/CR-6303 "Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems," documents several defense-in-depth and diversity analyses performed after 1990, and presents a method of performing such analyses that has been endorsed by the NRC in Staff Review Memorandum (SRM) dated July 21, 1993 to SECY 93-087. More recent guidance is provided in Digital Instrumentation and Controls (DI&C) Interim Staff Guidance (ISG) 02, Diversity and Defense-in-Depth Issues, Revision 2, June 5, 2009.

2. Properties of well formed requirements are as follows:

- a. Abstract
Each requirement should be implementation independent.
- b. Unambiguous
Each requirement should be stated in such a way so that it can be interpreted in only one way.
- c. Traceable
For each requirement it should be feasible to determine a relationship between specific documented statements and the specific statements in the definition of the system given in the FRS as evidence of the source of the requirement.
- d. Validatable
Each requirement should have the means to prove that the system satisfies the requirements.

3. Some pitfalls to avoid when building well-formed requirements are as follows:

- a. Desired and Implementation
There can be a tendency during the conceptual design stage to include 'features' or implementation decisions along with the requirements. Such information may still be important, but is not a true requirement.
- b. Overspecification
 - 1) Requirements other than performance specifications maybe overspecified. (For example, consider a race car, minimum speed might be a specified performance requirement, specification of engine size in this case is not relevant).
 - 2) Requirements that state tolerances for items deep within the conceptual system. Getting into detail beyond the system level.
 - 3) Requirements that implement solutions (requirements state a need).

CF2.ID9
ATTACHMENT 8.1

TITLE: Requirements Guidelines

c. Over-constrained

Requirements with unnecessary constraints. (For example, if a system must be able to run on rechargeable batteries, a derived requirement might be that the time to recharge should be less than 3 hrs. If this time is too restrictive and a 12 hr recharge time is sufficient, potential solutions are eliminated).

d. Unbounded

Requirements making relative, subjective, or vague statements. (For example, "user friendly," "cost effective," "quick response time," "highly reliable," "fault tolerant").

e. Assumptions

- 1) Requirements based upon undocumented assumptions. The assumption should be documented as well as the requirement.
- 2) Requirements based upon the assumption that a particular system or subsystem will reach completion. In this case the assumption and an alternative requirement should be documented.

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.2

TITLE: Functional Requirements Specification Outline

General

The Functional Requirements Specification (FRS) should contain the functional characteristics and performance criteria. The detail should be commensurate with the safety or financial risk significance of the system. The outline below is provided for consideration.

Content Outline

The FRS should be included as a drawing on the design change DDL. The engineer shall address as appropriate the following characteristics of the system in the FRS:

1. Introduction
 - System Purpose
 - System Scope
 - Definitions, Acronyms, Abbreviations
 - References
 - System Overview
2. General System Description
 - System Context
 - System Modes and States
 - Major System Capabilities
 - Major System Conditions
 - Major System Constraints
 - User Characteristics
 - Assumptions and Dependencies
 - Operational Scenarios
3. System Capabilities, Conditions, Constraints
 - Physical
 - System Performance Characteristics
 - System Security
 - Information Management
 - System Operations
 - Policy and Regulation
 - System Life Cycle Sustainment
4. System Interfaces

Refer to IEEE 1233-1998, "Guide for Developing System Requirements Specification"

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.3

TITLE: Software Requirements Specification Outline

General

The Software Requirements Specification (SRS) should contain critical characteristics and requirements associated with the software. The detail should be commensurate with the safety or financial risk significance of the system. The outline below is provided for consideration.

Content Outline

1. Introduction
 - Purpose
 - Scope
 - Definitions, acronyms, and abbreviations
 - References
 - Overview
2. Overall Description
 - Product perspective
 - Product Functions
 - User characteristics
 - Constraints
 - Assumption and dependencies
3. Specific requirements
 - External Interfaces
 - Functional Requirements
 - Performance Requirements
 - Design Constraints
 - Software System Attributes

Refer to IEEE 830-1998, "Recommended Practice for Software Requirements Specifications"

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.4

TITLE: Software Verification and Validation Plan Outline

Content Outline

1. Purpose
2. Referenced Documents
3. Definitions
4. V&V Overview
 - Organization
 - Master Schedule
 - Software Integrity Level Scheme
 - Resources Summary
 - Responsibilities
 - Tools, Techniques, and Methods
5. V&V Processes
 - Process: Management
Activity: Management of V&V
 - Process: Acquisition
Activity: Acquisition Support V&V
 - Process: Supply
Activity: Planning V&V
 - Process: Development
Activity: Concept V&V
Activity: Requirements
Activity: Design V&V
Activity: Implementation V&V
Activity: Test V&V
Activity: Installation and Checkout V&V
 - Process: Operation
Activity: Operation V&V
 - Process: Maintenance
Activity: Maintenance V&V
6. V&V Reporting Requirements
7. V&V Administrative Requirements
 - Anomaly Resolution and Reporting
 - Task Iteration Process
 - Deviation Policy
 - Control Procedures
 - Standards, Practices, and Conventions
8. V&V Documentation Requirements

Refer to: IEEE 1012-1998, "Standard for Software Verification and Validation"
IEEE 1059-1993, "Guide for Software Verification and Validation Plans"

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.5

TITLE: Problem Reporting and Corrective Action

PROJECT: _____

Page _____ of _____

Description: _____

By: _____ Date: _____ Critical ☐ Yes ☐ No

Corrective Action: _____

By: _____ Date: _____

Description: _____

By: _____ Date: _____ Critical ☐ Yes ☐ No

Corrective Action: _____

By: _____ Date: _____

Description: _____

By: _____ Date: _____ Critical ☐ Yes ☐ No

Corrective Action: _____

By: _____ Date: _____

Include as attachment to the project Software Verification and Validation Report (SVVR)

NUCLEAR POWER GENERATION
CF2.ID9
ATTACHMENT 8.6

TITLE: Software Configuration Management Plan Outline

General

The Software Configuration Management Plan (SCMP) defines a process driven approach for evaluation and mitigation of change-related risk. The intent of the SCMP is to document what configuration management activities are to be done, how they are to be done, who is responsible for doing specific activities, when they are to happen, and what resources are required. The detail should be commensurate with the safety or financial risk significance of the system. The outline below is provided for consideration.

Content Outline

1. Introduction
 - Purpose
 - Scope
 - Glossary
 - References
2. CM Management
 - Organization
 - Responsibilities
 - Risks
 - Security
3. CM Activities
 - Configuration Identification
 - Configuration Control
 - Configuration Status Accounting
 - Configuration Auditing
4. CM Schedules
 - Project Milestones
 - Acceptance Criteria
5. CM Resources
 - Development/Operating Environment(s)
 - Supporting Infrastructure
 - Automated Tools
 - Test Equipment
 - Training
 - Subcontractor Products
 - Vendor Products
6. CM Plan Management
 - Oversight

Refer to: IEEE Std. 828-2005, "IEEE Standard for Software Configuration Management Plans"