SOFTWARE PROGRAM MANUAL

FOR NUPLEX 80+

NPX80-SQP-0101.0

REVISION 01

ABB Combustion Engineering, Inc.
Nuclear Systems
Windsor, Connecticut 06095-0500

Issue Date: Nov. 10, 1993

RECORD OF REVISIONS

REVISION NO.	DATE	PAGES INVOLVED	PREPARED BY	APPROVALS
00	1/20/93	Original		
01	11/10/93	Changes on most pages		

TABLE OF CONTENTS

LIST OF EXHIBITS

LIST OF EXHIBITS (continued)

1.0 INTRODUCTION

1.1 PURPOSE

Computer software is essential to the design, analysis, operation and control of the Nuplex 80+ systems. This Software Program Manual (SPM) describes the requirements for the software design and development process and for the use of software in Nuplex 80+ systems. The SPM expands the procedural requirements for computer software in Reference 1.6.1.

The Software Program Manual consists of several basic elements:

1) A **Software Quality Assurance Plan**, which describes the process and practice of developing and using software. The SQAP addresses standards, conventions, reviews, problem reporting and other software quality issues.

2) A **Software Verification and Validation Plan**, which describes the method of assuring correctness of the software.

3) A **Software Configuration Management Plan**, which describes the method of maintaining the software in an identifiable state at all times.

4) A **Software Operations and Maintenance Plan**, which describes software practices after delivery to a customer.

The SPM also discusses Software Management, documentation and other matters related to software design and use.

Exhibit 1-1 depicts this document in relation to other documents for NUPLEX 80+.

1.2 SCOPE

1.2.1 This SPM shall apply to all software and firmware, whether developed in-house, licensed or procured from a commercial vendor, obtained from another organization or otherwise acquired and used in the Nuplex 80+ for delivery to a customer.

For the purposes of describing software methods, Nuplex 80+ systems and subsystems are identified as belonging to one of the following classes:

- Protection (safety critical)

  Software whose function is necessary to directly perform RPS, ESFAS and safe shutdown control actions.

- Important to Safety

  Software whose function is necessary to directly perform alternate protection system control actions or software that is relied on to monitor or test protection functions, or software that monitors plant critical safety functions as shown in Exhibit 3-1.

- Important to Availability

  Software that is relied on to maintain operation of plant systems and equipment that are critical to maintaining an operating plant.

- General Purpose

  Software that performs some purpose other than that described in the previous classifications. This software includes tools that are used to develop software in the other classifications, but is not installed in the on-line plant system.

Exhibit 1-2 lists typical assignments of Nuplex 80+ systems to these classes, although the assignments may vary according to specific contract requirements. Throughout the SPM, distinctions are made regarding the methods applied to each of the above classes. Specific parts of the software in a single system may be assigned to different classes. However, each part of the software must have an assigned class.

In addition to the system classification, the SPM makes distinctions regarding methods applied to each of the following categories of Nuplex 80+ software:

- Original, Developed for NUPLEX 80+
- Existing, to be Modified
- Existing, to be used as is

Every Nuplex 80+ software item is assigned to one of the above categories. In practice, software in several categories is included in each Nuplex 80+ system. For example, a typical computer system may rely on:

- An operating system from a commercial supplier that is existing, used as is.

- Some residual code to be updated from a previous project (existing, to be modified)

- New algorithms (originally developed)

Generally, this SPM applies to deliverable software. The SPM is also applicable to certain non-deliverable software items if the operation of other deliverable software depends on the adequacy and correctness of the deliverable item. For example, the SPM may be applied to a modelling program that generates test data for verification of a deliverable software item, unless that test data will be validated without inspection of the modelling program or its documentation. The cognizant engineering organization (CEO) shall determine which non-deliverable software items are to be treated the same as deliverable software per the SPM.

1.2.2 The following software is excluded from the requirements of this SPM:

- Administrative software and financial oriented application software such as accounting and business management software, used for purposes such as inventory, ordering, scheduling and project management.

- Commercial applications software for use in database management systems, word processing, and commercially purchased CAD systems. Such applications are Lotus 1-2-3, dBase, WordPerfect, and AutoCad.

1.3 OVERVIEW

Nuplex 80+ software developers shall proceed through a software development effort by following the approach described in this manual.

The software developers should first become familiar with the Software Quality Assurance Plan which is a guideline for all activities relating to Nuplex 80+ software.

The Cognizant Engineering Organization is required to determine the class and category of all software to be used for a specific system as described in the SQAP. Exhibit 1-2 is a typical determination of class and category for a system. The Cognizant Engineer identifies the applicable standards which must be followed for those specific classes and categories of software. The software tasks and responsibilities are outlined in the SQAP based upon software classification and category.

Each quality assurance task is described in the SQAP for each software life cycle phase. The narrative description, along with the corresponding Exhibit, assist the Cognizant Engineer in making the required decisions concerning the appropriate tasks to be performed and who is responsible for performing them. In addition, the specific documents which must be produced for each software life cycle phase are discussed in the SQAP. Required documents vary for each software category.

The Software Verification and Validation Plan, the Software Configuration Management Plan, and the Software Operation and Maintenance Plan describe the details of some of the activities outlined in the SQAP.

Adherence to the Software Verification and Validation Plan will ensure the verification of a faithful translation from one step in the software development process to the next step and the validation that the software product does fulfill the requirements for which the software was developed. The degree of independence required by this plan varies with the software classification. The applicability of the tasks varies with the software category.

The software configuration management plan describes the procedures necessary to maintain the Nuplex 80+ software in an identifiable state at all times. These procedures do not vary with the software class or category.

The Software Operations and Maintenance Plan describes the activities necessary to maintain the Nuplex 80+ software, to remove latent errors, to respond to new or revised requirements and to adapt the software to changes in operating environments. Activities for problem tracking for Nuplex 80+ protection class software shall continue through retirement. These activities associated with the other classes of software terminates at a time based on agreements with the customer. Responsibility to correct errors, to respond to new or revised requirements and to adapt the software to changes in operating environment are as defined contractually.

The documentation section of this Software Program Manual describes the various documents which are required. The minimum set of required documents for each software class is specified in the Software Quality Assurance Plan.

The problem reporting and corrective action section of the Software Program Manual describes procedures necessary to ensure that all software errors and failures are promptly acted upon and in a uniform manner encompassing all Nuplex 80+ software. The procedures in this section tie together the requirements of the Software Verification and Validation Plan and the Software Configuration Management Plan.

1.4 GENERAL REQUIREMENTS

1.4.1 The management and control of software applies to computer software and associated documentation developed or used for Nuplex 80+ applications. Software shall be developed, acquired, procured, controlled, and maintained in accordance with this software program manual

1.4.2 Software Life Cycle

This Software Program Manual is based on a Software Life Cycle model consistent with IEEE 610.12-1990 and ASME NQA-2a-1990 Part 2.7 which includes the following phases:

- Requirements Analysis
- Design
- Implementation or Coding
- Test
- Installation and Checkout
- Operation and Maintenance
- Retirement

1.4.3 Indoctrination and Training

As a minimum, Cognizant Engineering Organization personnel involved in Nuplex 80+ software design, development and use shall have documented training in this SPM. Such training records shall be prepared and maintained in accordance with the requirements of Reference 1.6.11.

1.4.4 Software Program Manual Control

Each copy of this SPM shall be numbered. The CEO shall maintain a list of controlled copy holders. Revisions including initial issue shall be approved by CEO management and Quality Assurance, as indicated on the cover sheet. Each page shall be sequentially numbered with the document title and revision level shown. Revisions shall be subject to the same controls as the original issue.

As a minimum a new cover sheet, record of revision page and revised pages shall be prepared for each revision.

This SPM shall be maintained by the CEO as a three year non-permanent quality record per the requirements of References 1.6.1 and 1.6.11.

1.5 ACRONYMS AND DEFINITIONS

ANSI	American National Standards Institute
CENSYS	Combustion Engineering Nuclear Systems
CEO	Cognizant Engineering Organization
CFR	Code of Federal Regulations
CM	Configuration Management
CMS	Configuration Management System
CPU	Central Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
I&C	Instrumentation and Control
NQA	Nuclear Quality Assurance
NRC	Nuclear Regulatory Commission
PDR	Preliminary Design Review
SCA	Software Change Authorization
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCR	Software Change Request
SDD	Software Design Description
SDR	Software Design Requirements
SOMP	Software Operation and Maintenance Plan
SPM	Software Program Manual
SQAP	Software Quality Assurance Plan
SRR	Software Requirements Review
SVVP	Software Verification and Validation Plan
TER	Test Exception Report
V&V	Verification and Validation

Baseline	Software which has been formally reviewed and that can be changed only through formal change control procedures.
Error	A deficiency in software which is judged to have the potential for significant impact on analysis results.
Computer Program	A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution as well as to execute it.

Configuration Item	A collection of hardware or software elements treated as a unit for the purpose of configuration control.
Configuration Control	The process of identifying and defining the configuration items in a system, and controlling the release and change of these items.
Cognizant Manager	Manager of the cognizant group responsible for control of a software configuration item. The Cognizant Manager may delegate the performance of necessary tasks to other persons but remains responsible for their execution.
Software	Computer programs and associated documentation and data pertaining to the operation of a computer system.
Software Life Cycle	The period of time that starts after a software product is conceived and ends when the software product is no longer available for use. The software life cycle typically includes a requirements phase, a design phase, an implementation phase, a test phase, an installation and checkout phase, an operation and maintenance phase, and sometimes a retirement phase.
Software Validation	The test and evaluation of the completed software/configuration item to ensure compliance with software requirements.
Software Verification	The process of determining whether the product of a given phase of the software development cycle fulfills the requirements imposed by the previous phase.

Software Verification and Validation Report	A document which provides objective evidence that a configuration item has been verified and validated for safety-related use.
Testing	The process of exercising or evaluating a system or system component by manual or automated means, to verify that it satisfies specified requirements or to identify differences between expected and actual results.
Test Case	A specific set of test data and associated procedures developed for a particular objective, such as to exercise a particular program path(s) or to verify compliance with a specific requirement.
Test Plan	A document describing the approach to be taken for intended testing activities. The plan typically identifies the items to be tested, the testing to be performed, test sequences, personnel requirements, and evaluation criteria.

1.6 REFERENCES

1.6.1 QPM-1.1 Quality Program Instruction

1.6.2 ASME NQA-2a-1990 Addenda, Part 2.7, "Quality Assurance Requirements of Computer Software for Nuclear Facility Applications".

1.6.3 (deleted)

1.6.4 (deleted)

1.6.5 (deleted)

1.6.6 (deleted))

1.6.7 (deleted)

1.6.8 (deleted)

1.6.9 NPX80-QPS-0401.1, Requirements for the Supply of Commercial Digital Computer Hardware and Software Components to be Used in Nuplex 80+ Safety Systems

1.6.10 Nuclear Engineering Documentation System

1.6.11 QPM-1 Quality Program Requirements

1.6.12 IEEE Std 610.12-1990, Standard Glossary of Software Engineering Terminology

1.6.13 IEEE Std 830-1984, Guide to Software Requirements Specifications

1.6.14 IEEE Std 1016-1987, Recommended Practice for Software Design Descriptions

1.6.15 IEEE Std 1012-1986 (Reaff. 1992), Standard for Software Verification and Validation Plans

1.6.16 IEEE Std 1063-1987, Standard for Software User Documentation

1.6.17 IEEE Std 828-1990, Standard for Software Configuration Management Plans

1.6.18	ANSI/IEEE-ANS-7-4.3.2, Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations, 1982
1.6.19	Quality Assurance Program Description, CENPD-210.
1.6.20	IEEE Std 730-1989, Standard for Software Quality Assurance Plans

2.0 ORGANIZATION

The ABB Combustion Engineering Nuclear Systems (CENSYS) organization includes the Instrumentation and Controls (I&C) Engineering department. The Nuplex 80+ systems development team is part of the I&C Engineering department.

Department management is responsible for the overall development and integration of Nuplex 80+ systems and associated software. The Nuplex 80+ cognizant engineering organization (CEO) is depicted in Exhibit 2-1. All Nuplex 80+ software development is managed by CEO managers who report to the I&C and Nuplex 80+ Engineering Director.

For example, a Manager may be assigned responsibility for one or more monitoring systems, control systems and/or protection systems. Supervisors of design groups report to the Managers. Each Nuplex 80+ system in the Manager's group of systems is the responsibility of a design group supervisor reporting to the Manager. This responsibility includes the development of software for the assigned system.

Members of a supervisor's group who perform software activities are organized into one or more of the following three teams:

- The **requirements team** develops functional, software design and performance requirements for systems that are assigned to their respective supervisory group, or in some cases to other groups.

- The **design team** develops or obtains the software for systems that are assigned to their respective group, based on requirements from the requirements team. In some cases, the design team may develop common software that is used in systems developed by other supervisory groups.

- The **verification team** performs verification and validation activities on systems assigned to other supervisory groups, for which the members of this group have had no requirements or design involvement. **This is an independence requirement.**

This verification team for a system reports to a group supervisor. The design team for that system reports to a different group supervisor. The requirements team for that system may be supervised by the design team supervisor, or another group supervisor other than the verification team

group supervisor. **This is an independence requirement.** Each group supervisor is responsible for the technical adequacy of the work done by the teams reporting to that supervisor.

Software personnel are not necessarily exclusively assigned to one of the above teams. A member of the supervisor's group may be on the requirements team for one system or function and the design team for the same or another system or function. The member may also be a member of the verification team for some system, provided that the independence requirement described above is met.

Managers and supervisors are assigned specific responsibilities and the authority to assure the accomplishment of software management and control through written plans, procedures, standards, and instructions.

Each of the CEO Managers is additionally responsible and accountable for:

- Nuplex 80+ plans, schedules, procedures, methods, and techniques required in the technical and administrative performance of the Nuplex 80+ related software within his responsibility.

- Compliance with the Software Configuration Management Plan.

Verification of the implementation of quality assurance requirements is performed by the independent CE Nuclear Systems Quality Assurance organization (see reference 1.6.11) according to 10CFR50, App. B requirements.

3.0 SOFTWARE QUALITY ASSURANCE PLAN

3.1 PURPOSE

3.1.1 Introduction

This Software Quality Assurance Plan (SQAP) describes the requirements and methodology to be followed by the CEO in developing, acquiring, using, and maintaining software to be used for the design and operation of software based systems for the Nuplex 80+ systems.

Software to be developed and used by ABB-CENSYS for the NUPLEX 80+ system shall be placed into the following software classes:

- Protection (safety critical)
- Important To Safety
- Important To Availability
- General Purpose

Exhibit 3-1 provides a typical assignment of software within these classes although actual assignment may vary based on specific contract requirements. Each group responsible for developing software shall document the appropriate class for each software module in its software design requirements and design description (see Section 3.7).

This SQAP requires system specific documents to describe in more detail the requirements for software configuration management, verification and validation and coding standards to be used on this project, as described in Section 3.2.2.1. This SQAP is not designed to duplicate information which will be contained in the system specific documents.

3.1.2 Scope

This SQAP is required for all quality classifications defined for the NUPLEX 80+ system: protection, important to safety, important to availability and general software within the scope of ABB-CENSYS software.

This SQAP is based on the software life cycle model which is described in Reference 1.6.2.

Within each software class described in Section 3.1.1, there are categories of software which this SQAP addresses. These categories are described as follows:

1) Original software

2) Existing software

   a. To be modified

   b. Not to be modified

All software modules developed or used within a system (see Exhibit 3-1) shall be documented by class and category using Exhibit 3-2. This shall be performed by the group responsible for the software (see Section 2.0). At least one sheet of Exhibit 3-2 shall be used for each class of software in the system.

Software that is initially assigned to one software class can be reassigned to any other class provided that all tasks appropriate for the new class, up to the current phase of the software life cycle, are completed and satisfactorily reviewed.

Existing software is software that has been created, but not under this SPM. To qualify for use under this SPM, the software must be evaluated by the CEO to meet the following criteria:

- Existing commercial software may be used for protection or important to safety software if it is qualified in accordance with reference 1.6.9. To qualify existing commercial software for important to availability and general purpose software, the CEO

shall select applicable portions of reference 1.6.9 and qualify the software to those portions.

- Existing NPP non-commercial software that has been actively used in a nuclear power plant may be used for the same class of software under this SPM, provided it has been maintained under an acceptable quality plan with an active program for problem reporting and corrective action reporting to the Nuplex 80+ project during the software life cycle. This software shall also have adequate design documentation, user documentation and well commented source code. This software shall have been verified and validated under another program that is judged by the V&V team to be acceptable.

- Other existing non-commercial software may be used if a review by the CEO design and verification teams concludes that it is well organized and has adequate design documentation, user documentation, and source code commentary.

For existing software that is qualified as above, design documentation and code may be used without revision to meet format or content requirements of this SPM. Modifications to this software may be made in accordance with prior documentation and code format. Qualified existing commercial software and NPP non-commercial software does not require verification of design documents and code. Other qualified existing non-commercial software requires verification of design documentation and source code in accordance with this SPM.

Under this SQAP, a software product that is contracted for development by a subcontractor is treated as original software unless the software already exists and is in use. In this case, it is treated as existing software.

This SQAP describes the methodology by which all software and associated documentation is managed throughout the life cycle. Software elements produced in the process of quality assurance are as follows:

- Test plans, cases, procedures and reports

- Review and audit results

- Problem reports and corrective action documentation
- Software configuration management plans
- Software verification and validation plans

3.1.3 Software Development Process

The software development process for original software is shown in Exhibit 3-3. This exhibit shows the relationship between software and hardware, the process of software integration and testing, the design documentation produced, and the quality assurance documentation required throughout the software life cycle.

As shown in Exhibit 3-3, software quality is assured through the process of verification reviews, validation testing at the different stages of development, and software configuration management during all phases of software development. Software Verification and Validation (V&V) activities are governed by system specific Software V&V Plans. Required test procedures and test reports are shown in Exhibit 3-3, and are based on the level of the test and the class of the software.

3.2 MANAGEMENT

The management of all software for the Nuplex 80+ project spans the software life cycle defined in Reference 1.6.2 and applies to all software classes described in Section 3.1.1.

3.2.1 ORGANIZATION

The implementation of an effective SQAP is the responsibility of all persons involved in the software development process. Each person responsible for the software development shall perform their work in accordance with established standards, methods, and procedures identified in this SQAP.

Software life cycle activities for this project shall be performed by the Cognizant Engineering Organization (CEO) described in section 2. The CEO is responsible for the execution of all quality assurance tasks including Verification and Validation (V&V) and software configuration management.

Management of the SQAP is overseen by the CEO Manager. Organizationally, verification of the implementation of quality assurance requirements is performed by Quality Assurance in accordance with References 1.6.1 and 1.6.11.

The CEO shall ensure that software and associated documentation has been developed in accordance with the standards specified in this SQAP. This includes ensuring that the coding standards (Section 3.4.2.1), testing standards established in the test plan and documentation standards (Section 7) have been followed.

In general, software configuration management responsibilities for the CEO span all phases of the software life cycle for

- Development of Software Configuration Management Plans
- Execution of software configuration management activities per the SCMP
- Control of software through a librarian
- Baselining and integration of new software versions

3.2.2 Tasks and Responsibilities

This section describes the specific tasks and responsibilities to be performed by the CEO. All tasks and responsibilities described in this section apply to each system assigned to the CEO. Tasks are listed in the life cycle phase for which they will be performed. Typical tasks are: software design and development, software quality assurance planning, verification reviews, audits, test planning, test execution, and test reporting. Tasks required are based on software category. Exhibit 3-4 shows the software tasks for each category in each phase.

The following are some procedural type of actions that are performed to ensure traceability throughout the development and verification stages:

1) The project system description and design specification documents are dated and signed by the designer and the design team manager.

2) Each original or revision of a software module written by a programmer is dated and signed by the programmer and the lead engineer.

3) The corresponding project software verification report and software test procedures documents are dated and signed by the author and the verification team group supervisor.

4) Each software module is verified, dated, and signed by the verifier.

3.2.2.1 SQA Planning Phase

During this phase, system specific software quality assurance planning (if needed) shall be performed. This includes system specific Software Verification and Validation Plans (SVVP's) and Software Configuration Management Plans (SCMP's). Also coding standards shall be developed by the CEO.

System specific SVVP's shall be developed by each CEO group responsible for delivering software. System specific SVVP's shall be developed in accordance with Section 4 of this SPM.

System specific SCMP's shall reflect the specific methods of managing the software configurations within the group responsible for delivering the system(s). These SCMP's shall address software configuration management activities for all classes (and categories within classes) of software. The system specific SCMP's shall be developed in accordance with Section 5 of this SPM.

Requirements for the development of documents shown in this phase are detailed in Section 7.

3.2.2.2 Functional Requirements Phase

During this phase, System Functional Requirements are developed. These documents shall provide the necessary detail to develop software and hardware requirements. The hardware specifications shall be for both the software development platform and the system deliverable hardware (see Exhibit 3-3). These system functional requirements are noted in Exhibit 3-4.

During this phase, prototype software may be developed to prove a new principle or to help further define the functional design. As such, prototype software has a different (and usually shorter) software life cycle than the other categories of software. Specifically, prototype quality assurance tasks shall include:

- Adherence to coding standards
- Documentation of prototype design (format at the discretion of the individual CEO group)
- Informal verification reviews within the CEO group
- Limited software configuration management

During the development of prototype software, the individual CEO group responsible should stress the importance of its reuse potential into the deliverable system software.

In the case where prototype software is reused and integrated into the deliverable software, it shall undergo the respective software quality measures based on its software class. This includes software quality assurance tasks described above from the integration point forward in the

life cycle plus any "skipped" tasks in the life cycle for; verification reviews, audits, and software configuration management activities required documentation and conformance to coding standards.

A system specific test plan shall be developed based on the SVVP and identify how the test activities will be implemented. It shall include the following topics as a minimum:

- General approach including: identification of test procedures and test cases, general test methods, documentation of results, and traceability methods to the SDR and SDD.
- Requirements for testing including: test boundary conditions on inputs and unexpected input conditions.
- Test management including: personnel, resources, organization, schedule and responsibilities.
- Procedures for qualification and control of the hardware to be used in testing.
- Qualification and use of software tools.
- Installation test requirements for existing software which is used without modification.
- Regression test requirements for existing software which is modified.

Test plan development continues into subsequent phases and is completed in the implementation phase.

3.2.2.3 Software Requirements Phase

The CEO shall be responsible for developing, maintaining, and updating its Software Design Requirements (SDR). An SDR shall be developed for each system based on system functional requirements, and shall provide the detail and information sufficient to design the software. For systems which have functions spanning more than one software class (see Exhibit 3-1), the SDR shall be divided to describe software requirements for the software in each class. For example, the SDR for the Data Processing System (DPS) shall differentiate software requirements for the Safety Parameter Display

System (SPDS) functions from software requirements for the rest of the DPS. The SDR shall be developed in accordance with Section 7.3 of this SPM.

Each SDR shall be verified by the V&V team or Functional Requirements Team, as shown in Exhibit 4-1. The verification review shall ensure that the functional requirements identified in the System Functional Requirements are properly reflected in the SDR. Verification of SDRs shall be performed in accordance with Section 3.5.2.1.

3.2.2.4 Software Design Phase

The CEO shall be responsible for developing, maintaining and updating a Software Design Description (SDD) for each software module. Each SDD shall be traceable to the requirements set forth in the SDR, and shall include enough detail to begin coding in the Implementation Phase. All SDD's shall be developed in accordance with the requirements of Section 7.4.

Each SDD shall be verified by the teams indicated in Exhibit 4-1. The verification review shall ensure that the software requirements identified in the SDR are properly reflected in the SDD. Verification of SDD's shall be performed in accordance with Section 3.5.2.2.

3.2.2.5 Software Implementation Phase

Original software development and modifications to existing software shall begin with module coding by the CEO in accordance with the appropriate coding standard(s) listed in Section 3.4.2.1.

Existing software which has been qualified as described in Section 3.1.2 may be integrated into the software system and tested during this phase.

Verification of module code listings shall be performed by the group identified in Exhibit 4-1. Details of software module code verification is described in Section 3.5.2.3.

The test plan, started during the functional requirements phase, is completed in this phase.

Testing during this phase can be accomplished by several methods at the discretion of the CEO group. Some possible methods are identified below:

- One method is to hierarchically assemble the modules into units and perform a unit test, and subsequently assemble all the units into the system and perform a system test.

- Or, the test sequence can be performed in a series of expansions. This could be accomplished by continually adding successfully tested modules to the "system" and test after each addition until the complete system is assembled and tested.

Module and unit testing shall be performed in accordance with the Test Plan. The responsibility for testing will be assigned to the design team and V&V team as shown in Exhibit 4-1. Unit test procedures and reports are only required for software classified as protection and as important to safety.

After successful unit testing, the software shall be certified as "Unit Level Certified" by the CEO group. Certification means that the software has been tested in accordance with the Test Plan, all code reviews have been completed, and all errors have been resolved or documented. Exhibit 3-5 shall be used to document the unit test certification.

3.2.2.6 Testing Phase

System testing shall be conducted, per the Test Plan, during this phase in the development environment when all of the system components have been integrated by the CEO group. The purpose of this test is to evaluate the system as a whole for its ability to meet system usage and performance requirements. Test procedures and reports shall be documented and verified by the groups identified in Exhibit 4-1. System tests shall be conducted by the groups identified in Exhibit 4-1. Inter-system testing (the process of testing system interfaces) shall be performed in accordance with the Test Plan. Also, test procedures and reports shall be developed consistent with the Test Plan.

After successful system testing, the software shall be certified by the V&V team as: "System Level Certified". After successful inter-system testing, the code shall be certified by the V&V team as: "Inter-System Level Certified". If the software undergoes and passes control complex integration testing, it shall be certified by the V&V team as: "Control Complex Level Certified".

All User Documentation shall be developed during this phase in accordance with Section 7. Also, during this phase, all user documentation shall be verified by the V&V team.

3.2.2.7 Site Installation and Checkout Phase

During this phase the software becomes part of the installed equipment incorporating applicable software components, hardware, and data. The process of integrating the software with applicable components may consist of installing hardware, installing the program, reformatting or creating databases, and verifying that all components have been included.

Site installation and checkout activities shall be performed by the site startup group. An installation log shall be maintained during the installation and checkout process. This log shall be verified by the V&V team after installation.

After installation, the equipment and software shall be checked out according to the test plan and Site Acceptance Test (SAT) Procedure. All SAT reports shall be documented. The preparation of the test plan and procedures will be initiated during the requirements phase, to support evaluation of requirement testability. The test procedure shall be verified by the V&V team. Verification of the installed software shall be performed to determine that the software was installed correctly. This shall include checksum verification and some limited functional testing. Software installation verification applies to initial software and any subsequent revisions.

In this phase, the Software Verification and Validation Report (SVVR) shall be prepared. Details of the SVVR are described in Section 7.6.

3.2.2.8 Operations and Maintenance Phase

Activity in this phase consists of maintenance of the software to:

- remove identified latent errors
- respond to new requirements, or
- adapt the software to changes in the operating environment.

Software modifications shall be approved, documented, verified and validated, and controlled in the same manner as described previously in the Design, Implementation and Test Phases. Each system specific SVVP, in conjunction with the SCMP, shall also be used to assist in the management of these activities and procedures.

3.3 DOCUMENTATION

3.3.1 Purpose

Documents required by this SQAP are listed in Exhibit 3-4. Section 7 of this SPM provides guidance for the development of documents. All documents listed shall be made lifetime quality records in accordance with the requirements of Reference 1.6.11.

3.4 STANDARDS, PRACTICES AND CONVENTIONS

3.4.1 Purpose

The standards, practices and conventions to be applied to the Nuplex 80+ project are contained in Reference 1.6.1. Compliance with these standards shall be monitored and assured via the SQAP and system specific documents.

3.4.2 Contents

3.4.2.1 Coding Standards

The software development process shall provide guidance to ensure standardization, compatibility and maintainability of resulting software products. The process shall provide a coding standard for each language, database, or software tool that allows author discretion in establishment or use of convention.

This requirement applied to the following typical software products:

C/C++	Assembly languages
Fortran	Database query languages
Relay Ladder Logic	Display building languages

Each coding standard shall contain, but is not limited to, the following information:

1) General

This area outlines general ideas and concepts used to guide the creation of software written under a specific language.

2) Naming conventions

- Filename extensions as far as how they are used to organize files.
- Information pertaining to file organization within a system.
- Variables naming.

3) Internal documentation guidelines

- Program identification header content, placement, type, quality, and quantity.
- Revision history recording within each source file.

4) Stylistic conventions - issues that affect readability, such as indentation and use of white space.

5) Use of specific Language features

6) Software tool usage guidelines

- Information and use of automatic make facilities
- Appropriate compiler flag usage.

7) Functions

- Modularity
- Naming

3.4.2.2 Software Testing Standards

Software testing methodologies, policies and practices shall be described in the system specific Test Plan. Specific format and content for test procedures, test cases, and test reports shall also be provided in the Test Plan.

3.4.2.3 Documentation Standards

All documents developed for the Nuplex 80+ System shall comply with the requirements for format and content described in Section 7.

3.5 REVIEW REQUIREMENTS

3.5.1 Purpose

The purpose of this section is to address the review requirements throughout the software life cycle.

The software reviews required by this SQAP address software classes and categories described in Sections 3.1.1 and 3.1.2.

Reviews are technical in nature and are designed to verify the technical adequacy and completeness of the design and development of the software.

Review activities applicable for original software are:

- SDR verification,
- SDD verification,
- code verification, and
- program documentation verification including each SVVR and testing review.

The reviews are the responsibility of the individual CEO group as described in Exhibit 4-1. The methodology of performing reviews shall be included in each system specific SVVP or Section 4. Reviews performed within the CEO group shall be performed by peers who have an equivalent knowledge of the topic but who are not directly involved with the application as required in Section 2.0.

Audits are designed to ensure that software documentation and processes comply with the established standards and guidelines set forth on the project.

References to the SVVP are provided in this section to address system specific areas of the review and audit process. The intent of this section is to provide guidelines to conduct reviews and audits. Procedural aspects of the review shall be contained in the SVVP.

3.5.2 Requirements

Technical reviews shall evaluate specific software elements (such as files, functions, subsystems, or complete systems) to ensure that the requirements are adequate, technically feasible and complete. The technical review shall be managed by the CEO design group according to the SVVP, and shall address the following items prior to all reviews:

- Statement of objective for the technical review
- Software element(s) to be examined
- Specifications for software elements to be examined
- Plans, standards or guidelines against which the software elements are to be examined

3.5.2.1 Software Requirements Review (SRR)

The Software Requirements Review (SRR) shall examine the Software Design Requirements (SDR) to verify that it is clear, verifiable, consistent, modifiable, traceable, and usable during the operations and maintenance phases. The SRR shall include evaluation of the software requirements against the user's software application which is described in a higher level requirements document such as a functional requirements document.

Specific SRR items are described in Section 4 and shall be described in detail as necessary in the SVVP. As a minimum, these items shall include:

- Traceability and completeness of the requirements
- Adequacy of rationale for derived requirements
- Testability of functional requirements
- Adequacy and completeness of verification and acceptance requirements
- Conformance to documentation standards
- Adequacy and feasibility of performance requirements
- Adequacy and completeness of interface requirements

Responsibilities, methodologies, and reporting of results are described in Section 4 and shall be described in detail as necessary in the SVVP. Frequently encountered categories or types of errors normally found in the SDR may also be included in the SVVP in order to aid the independent reviewer.

3.5.2.2 Software Design Review

The Software Design Review shall determine the acceptability of the detailed software design as depicted in the Software Design Description (SDD) in satisfying the requirements of the SDR.

Software design review items are described in Section 4 and shall be described in detail as necessary in the SVVP. The Software Design Review shall be performed at the completion of the SDD and generally address the following items:

- Technical Adequacy
- Completeness of design requirements
- Consistency
- Correctness of software design
- Traceability of software design to software requirements

Responsibilities, methodologies, and reporting of results shall be described in detail in the SVVP. Frequently encountered categories or types of errors normally found in the SDD may also be included in the SVVP in order to aid the independent reviewer.

3.5.2.3 Code Verification

Software code shall undergo periodic peer review by means of a code inspection. The exact methods and frequency of this review shall be specified in each system SVVP.

Code reviews are the responsibility of an independent reviewer within the CEO as shown in Exhibit 4-1, and shall be performed in accordance with the Software Coding Standards described in Section 3.4.2.1. Code reviews shall include evaluation of the source code implementation against the SDD.

3.5.2.4 Software Verification and Validation Report Review

The Software Verification and Validation Report Review shall be performed to evaluate the adequacy and completeness of the SVVR. The review shall verify that all reviews and tests have been properly documented in the SVVR. The review shall also verify that the level of independence in each review and test has been maintained in accordance with Exhibit 4-1. This review shall be performed by an independent reviewer not associated with the development of the system specific SVVP (see Section 7.6). The review shall be conducted at the end of each phase in the software life cycle, and documented on a Comment Record form(s) Exhibit 3-6.

3.5.2.5 Testing Review

The Testing Review shall be conducted to ensure that the software testing activities are performed and documentation prepared in accordance with the SVVP and its supporting test plans, cases, procedures, and reports. The Testing Review shall be performed by an independent reviewer at the conclusion of the Test Phase, and be documented on a Comment Record form(s). (Exhibit 3-6)

The testing review process is described in Section 4 and shall be documented in detail as necessary in the SVVP.

3.6 PROBLEM REPORTING AND CORRECTIVE ACTION

3.6.1 Purpose and Scope

The purpose of a formal procedure of software problem reporting and corrective action is to ensure that all software errors and failures are promptly acted upon and in a uniform manner encompassing all project software. This procedure ties together the requirements of the SVVP and the SCMP. V&V activities are the primary vehicle to uncover software problems, while the SCMP shall ensure that actions taken to correct problems by changing configured software are consistent and traceable.

Problem reporting and corrective action procedures shall span the entire software life cycle and all software classes identified in this SQAP. These procedures are detailed in Section 8 of this SPM.

3.7 TOOLS, TECHNIQUES, AND METHODOLOGIES

Software development for NUPLEX 80+ shall use a number of techniques to help assure all software is designed, implemented, and documented in accordance with the NUPLEX 80+ objectives of building software which meets the requirements and which is maintainable over time in the most cost effective manner. The tools, techniques and methodologies employed in this process shall ensure that the software is verifiable from each phase of the project to the next.

- Use of structured design techniques for analyzing and developing the software design. These shall include data flow diagrams, where applicable, to represent the interactions among modular elements and the flow of data among them. Entity-relation charts may be used to represent any relational database structures.

- Management sign-off and approval of all design and V&V documentation shall include the immediate supervisor and manager of the preparer. In addition, any affected supervisor(s) of the document shall be included in the approval process.

- Use of the spiral model of software development and testing techniques to help assure that the interfaces between various software products are well-isolated and tested. In some cases, the structured (waterfall) model of software development may be more appropriate. Also, bottom-up testing may be incorporated where applicable.

- The use of commercially available automated tools for software configuration management should be employed to the maximum extent possible.

3.8 CODE CONTROL

The purpose of this section is to describe the methods and facilities used to maintain, store, secure and document controlled versions of the identified software during all phases of the software life cycle.

3.8.1 Software Configuration Management

3.8.1.1 Purpose

The purpose of software configuration management is to:

- Maintain historical records of software as it is developed,used,and delivered and its relation to functional documents,
- Control changes to the software, and
- Record and report change processing and implementation status.

All software items and associated documentation shall be controlled to maintain the items in a known and consistent state at all times. Original software and modifications to existing software shall follow the configuration requirements for all life cycle phases. Existing software which is not modified shall only fall under configuration control procedures upon its introduction into the software system.

A Software Configuration Management Plan (SCMP) is described in Section 5 of this SPM. The SCMP shall provide the necessary means to enable traceability of software and documentation products following their origination.

3.9 MEDIA CONTROL

The methods and facilities used to protect computer program physical media from unauthorized access or inadvertent damage or degradation are described herein.

3.9.1 Media Identification

Each removable disk pack, floppy disk, or tape cartridge shall contain a serial number and description which lists the file(s) backed up on the medium. A log shall be maintained which cross references serial numbers to file(s) backed up. Disk packs, floppies and tape cartridges should not be switched, renamed, or initialized without prior approval from the CEO group.

A tape rack (or cabinet) and disk file cabinet shall be provided for storage and easy access.

3.9.2 Archival Requirements

One locked fire/water proof vault shall be installed and maintained at a separate premise from the computer facility to store all project software. This vault shall be able to accommodate the storage of all utilized types of media, such as:

- 2400 foot magnetic tape reels
- 5.25" and 3.5" floppy disks
- magnetic tape cartridges

After important NUPLEX 80+ software development milestones or baseline configurations are archived, a known software configuration shall be completely backed up and stored in the vault.

3.10 SUPPLIER CONTROL

The purpose of this section is to describe the level of software quality assurance measures to be applied to software supplied to the NUPLEX 80+ system from parties other than the CEO.

3.10.1 Existing Software

This SQAP defines existing software as software which was previously developed prior to the NUPLEX 80+ system to satisfy a general market need and may be considered for use on this project. The software may be subsequently modified prior to delivery, or it may be used "as is".

Existing software includes commercial software which is integral to the delivered system and software which is determined to be in support of the delivered system. Examples of integral software would be:

- Operating systems
- Compilers, Linkers, Loaders
- Database software
- Communication Drivers
- Man-Machine Interface software
- Display building software

All commercial software which will be used in Nuplex 80+ safety systems must meet the requirements established in Reference 1.6.9.

For existing software which is modified prior to delivery, all software requirements specified in this SQAP for original software shall be in effect for at least the modifications, except that modifications may conform to coding standards already in use. The minimum V&V activities applicable for modifications to existing software are: software modifications requirements verification, software modifications design verification, program modification documentation verification, and software validation. Regression testing using test cases shall be conducted to ensure that the modifications do not produce unintended adverse effects, and to ensure that the modified software still meets the original software requirements.

Existing software that is not modified shall be qualified for use according to Section 3.1.2.

Once qualified for use, the software shall fall under the system specific SCMP. Once installed, the software shall meet the following requirements:

- Verification and Validation during Installation and Operation per the SVVP,
- Configuration Management during Installation and Operation per the SCMP,
- Documentation including: Test Plans, Procedures, SVVR, and User Manuals,
- Problem reporting and corrective action procedures, and
- Records of delivered documents and software

3.10.2 Sub-Contracted Software/Services

Original software which is developed by a contractor and purchased by ABB-CENSYS shall adhere to the quality assurance requirements specified in this SQAP for original software. This applies regardless of whether the software will be subsequently modified or not.

Where available, third party contractors who supply software are required to provide feedback to ABB-CENSYS of problems encountered by the supplier or other users of similar software. Where available, this feedback information shall be supplied to ABB-CENSYS automatically without request from ABB-CENSYS. Also, each CEO group has the responsibility of informing the supplier of such software of any problems encountered by the user in the use or maintenance of the software.

Additional requirements for subcontracted software and services are as follows:

- Software and services must be procured from approved supplier per Reference 1.6.11
- Suppliers must have written quality assurance policies which meet the principles and intent of this SQAP.

- Purchase orders shall require the supplier to make available documents which are evidence of compliance with the principles and intent of this SQAP.

- Purchase orders shall require the supplier to deliver adequate user documentation, test procedures and test reports.

4.0 SOFTWARE VERIFICATION AND VALIDATION PLAN (SVVP)

4.1 PURPOSE

The purpose of this section is to establish requirements for the V&V process to be applied to Nuplex 80+ systems. The section includes various V&V methodologies aimed to increase the system reliability and availability. Some of these methodologies employ systematic checks for detecting errors in the system hardware and software, during the system development and implementation process. This section explains requirements for the V&V processes starting with the system design document stage and all necessary V&V activities to verify and/or validate I&C systems. This SVVP complies with ANSI/IEEE ANS 7.4.3.2 (1982).

The goals of this V&V plan, when applied to a specific project, are to:

- Improve the system reliability and availability
- Reduce system costs by exposing errors as early as possible
- Provide a systematic process of objectively evaluating the system's performance
- Demonstrate compliance with customer requirements, industry standards and licensing requirements.

4.1.1 Meaning of Verification

Verification is the process of checking the results of the stage-by-stage system development to determine that there is a faithful translation of one stage (such as design) into the next stage (such as implementation).

4.1.2 Meaning of Validation

Validation is the process of determining the conformance between the operational system and the system requirements, under normal and abnormal simulated plant operational conditions. Thus, it provides the overall assurance that the capabilities specified in the system requirements are implemented in the hardware and software, and the system is properly integrated. Another goal is to ensure that problems or potential deficiencies that may have occurred during the design and implementation phase have been corrected.

4.1.3 V&V Program Implementation

V&V activities are integrated into the requirements, design, implementation, test and installation phases described in Section 3. Experience has shown that the earlier a deficiency is discovered, the easier and more economical it is to resolve. The initial activity is the review of system functional requirements prior to any detailed design of hardware or software. Verification activities are performed at the end of this phase, and each subsequent phase. These activities determine that all requirements have been properly transferred from the input products to the output products of the phase, with amplifications or modifications appropriate to the phase. Upon completion of the software implementation, validation activities are performed. These activities determine that the operation of the system is consistent with the system requirements. Thus V&V activities are integrated with project activities from the beginning to end.

Once a system design and implementation has been verified and validated, any succeeding systems manufactured of the same design are certified by standard manufacturing test procedures. Many of the tests used by manufacturing are the same or equivalent to those used in the system V&V process. The equivalent of the system validation tests are performed as a minimum on every successive system of the same design that has been previously verified and validated. This is referred to as a Factory Acceptance Test. Traceability of all tests performed on manufactured units are maintained under configuration management control. Any design changes that would impact manufactured units are reverified and maintained under configuration management control.

4.1.4 System Class and Reviewer Independence

All Systems are classified as protection, important to safety, important to availability, or general purpose as described in Section 1. V&V activities vary according to system class in the degree of independence required for the reviews. Protection systems utilize non-complex software items and review of any particular protection software item shall be performed by a person who is competent and organizationally independent of the system design.

Organizationally independent means that the reviewer has not been a member of the system's requirements team or design team while the system has been designed. It further requires that the reviewer reports to a group supervisor other than the supervisor of the requirements team or

design team who developed the software. Such independence is important to obtaining an objective review, based solely on the documents and materials produced, and not influenced by designer intentions or assumptions. Consideration shall be given to performance of these reviews by outside agencies or consultants.

The reviewers of software in non-safety critical classes may be members of the requirements team or, in some cases, the design team. Nevertheless, the review of any particular software item shall not be performed by the individual(s) responsible for the requirements or design of the item. An independent reviewer must also be one who can perform a competent review.

Exhibit 4-1 identifies the minimum review independence required for each type of document or software item, for each class of system.

4.1.5 Categorization of Software Items and Review Scope

V&V is performed on documents and materials that are produced according to the category of each software item, as described in Section 3. For example, a software design description is not required for an existing commercial off-the-shelf software package. V&V activities only include documents and materials identified in Section 3.

4.1.6 Prominence of V&V Documentation

Traceability is important, not only to document the V&V activities, but also to record appropriate actions taken to resolve discrepancies. Thus a V&V program is, by its nature, oriented heavily towards documentation and the ability to trace changes in project documents. The generic documentation plan defines the structure and format of contents of the documents which may be produced during various phases of the project. The documents contents will vary depending on the specifics of system or project; however a system to trace the documentation and deficiency resolution is required. In the early phases of the system design process the system is divided into manageable modules of software and hardware. In the later phases, these modules are integrated into a total system.

The Configuration Management Plan addresses these issues and details (1) how the documents are controlled, (2) how records of changes and distribution are maintained, and (3) status of each document is identified.

4.1.7 Overall Nuplex 80+ and System Specific V&V Plans

This overall Nuplex 80+ V&V plan details the V&V process and activities involved during the various phases, and details various tools and techniques that can be used. System specific V&V details are developed before the start of system design, including schedule, reso rces to be used, titles of documents to be produced, and software item category and classifications.

4.1.8 Requirements Traceability Matrix

Throughout the project life-cycle, a software requirements traceability matrix will be created and maintained for each system.

The matrix provides a method to cross reference each software requirement against all of the documents and other software items in which it is addressed. The purpose of this matrix is to assist the reviewer in determining the propagation of requirements through the software items.

A typical format of the Matrix (see Exhibit 4-9) lists all of the requirements down the left side, where each row is a separate requirement. All documents and other software items are listed across the top, where each column is a unique document. The items listed across the top of the matrix include revisions of every item. The inclusion of revision documents within the matrix provides a history of requirements changes throughout the project. A database format is acceptable provided its contents include the information and relationships described in this section.

Requirements entered in the matrix are organized into successive sub-requirements as described in each document.

Within each cell of the matrix, a code is entered to denote the status of the requirement within the document. The codes are as follows:

Y - Requirement is included in this document
N - Requirement is not included in this document

The V&V team shall revise the traceability matrix throughout software development. The V&V team will review the matrix whenever it is changed to determine that requirements are appropriately included in software documents and their revisions.

4.2 OVERVIEW OF V&V

4.2.1 Verification and Validation (V&V) Principles

This section describes a number of concepts associated with the V&V process. In general, they cover the V&V activities during the system development and implementation process and will affect one or more stages of the system development/implementation.

4.2.1.1 Functional Requirements Documents

1) The functional requirements documents, which are the first product in the system development life cycle, are required for any verification program.

2) The functional requirements documents, as the name implies, explicitly state what the system is required to do.

3) Requirements documents are written by the system designer to establish the general limitations of the system scope and performance.

4) Requirements documents are also written by other to impose specific requirements for the system to meet.

4.2.1.2 Successive Congruence of Document Revisions

1) Design specifications shall be written which are both unambiguous and complete.

2) Each verification process establishes traceability between revisions of successive document.

3) A systematic method for carrying out this traceability shall be included in the configuration management scheme.

4) The V&V process shall employ the traceability matrix described in Section 4.1.8.

4.2.1.3 Implementation of V&V Through System Life Cycle

1) The V&V process is integrated into all phases of the system development life cycle rather than isolated into a separate testing stage which takes place long after the design is developed.

2) Informal reviews and other verification methods (described in Section 4.3) take place throughout each development phase.

3) Throughout the development process and during the formal system design reviews, the intermediate and final products are reviewed and evaluated.

4.2.1.4 V&V In Parallel with Design Activities

Verification is to be performed in parallel with the design efforts in order that errors may be detected and corrected as soon as they are made.

4.2.1.5 Configuration Management and V&V

The correctness of a delivered software configuration greatly depends on the care with which its configuration has been managed. Lack of care in this area can result in the improper release of outdated or premature versions of code. To protect against such problems, V&V activities shall also include the review of configuration management of each software item. This review shall assure that each software item has an identifiable design status and release history, as well as records or revisions and attributable problem reports. Releases of software items shall be consistent with an identified set of revisions of design documents. No software item may be released without approval by the V&V team.

Configuration management shall be performed as described in Section 5.

4.2.1.6 Change implementation and V&V

V&V activities shall be performed on both original software items and revisions. If revisions are significant, V&V is performed as though an original item were developed. Minor changes may require limited V&V, perhaps only the part of the software item modified. The scope of the V&V reverification is determined by the V&V team.

4.2.1.7 Automated Tools

Part of the V&V planning process includes the selection of appropriate tools for a given project.

4.2.1.8 V&V Core Activities

The following V&V core activities are applicable to every system (irrespective of the selected verification test strategy).

1) Upon completion of the V&V review of a particular software item, the reviewer will complete and sign the checklist (Exhibit 4-2 through 4-8) for the phase in which the preparation of the software item is completed. The questions in this checklist provide a basic set of considerations that the V&V reviewer shall include in the review.

2) Reviews assure clear, accurate and complete documentation detailing the system, hardware and software design requirements and design specifications.

3) System validation testing, as a minimum, will be performed on every system as part of both the development and manufacturing processes; details of test bed, validation test procedures and test results will be documented for review and audit purposes.

4) Unit Testing will be performed by the design team or V&V team according to Exhibit 4-1. The test plan, test data base and test results will be documented, as required. Note: There is not necessarily a one-to-one relationship between design units, implementation units and test units. Several design units may make up an implementation unit (e.g. a software program) and several implementation units may make up a test unit.

5) The V&V team will participate in design reviews, audits and configuration management activities.

4.2.2 The V&V Process

The top-down design decomposition process begins with the development of the requirements documents in establishing the overall system design.

1) Input information for the system design is provided by the functional requirements document that defines the system functional operations.

2) Accompanying input documents shall involve the quality assurance requirements, and if applicable, man-machine, fluid systems, and balance of plant requirements.

3) Although formal design reviews, which can be used as part of the V&V process, may occur earlier, the initial V&V stage conducted by the V&V team is done after the functional requirements are completed.

4) The system design documents are then verified back to the functional requirements document as the major reference document.

5) Guidance for this stage of verification is provided in the hardware, software, and system verification requirements.

6) Hardware, software and system verification requirements are written as necessary by the verification team, independent of the design team which conducts the system design.

7) The method for reporting any discrepancies found at this stage, or at any other V&V stage, is explained in procedures established by the problem reporting and corrective action (Section 8).

8) A second stage of verification follows the establishment of the software design description, whereby the software design description is verified back to the preceding system design step and the functional requirements documents.

9) The module implementation documents which provide the descriptions of the hardware and software are produced during the next step of the development and implementation process.

10) Specifications, which may include electrical and mechanical assembly drawings, rack and cabinet internal wiring diagrams, and internal power distribution diagrams furnish the descriptions for the system hardware implementation.

11) System software implementation is provided by the source program and the object program.

12) A third stage of verification follows, which involves the module implementation verification by reviewing the implementation and by also reviewing the actual physical testing of the modules. Module testing is conducted by the design team.

13) Hardware validation test procedures and software validation test procedures are provided for the validation process. These are developed and reviewed by the requirements, design and V&V teams according to Exhibit 4-1.

14) All validation testing of the system software and hardware modules is conducted by the requirements, design or V&V team according to Exhibit 4-1.

15) A report of the hardware test results and the software test results is provided after this stage of verification.

16) These reports in conjunction with the hardware, software, and system design documents, and the validation test procedures, contain sufficient information to enable a third party to repeat any of the tests and understand the test results.

17) The system integration is performed during the next step of the implementation phase.

18) The system integration verification and validation stage shall follow this final step of the system development and implementation process.

19) The writing of documents to perform the system verification and validation test, and the conductance of the test are under the control of the V&V team. Participation in these efforts by the design team members is permitted, as long as the designer was not directly responsible for that portion of the system to be tested.

20) Any environmental, temperature and humidity validation tests are conducted before the end of this V&V stage if such tests are required.

21) A test results report is compiled at the conclusion of the tests.

22) Summaries of these test results and the summaries of the test results compiled after the unit validation tests are available for review as part of the licensing process.

23) Provisions for document design changes, deletions, or additions to the system shall be included in the system maintenance manuals. Any revisions of the system hardware and software designs shall be contained in these documents, as well as any required documentation specified by the configuration management program that may relate to changes that would require reverification.

24) The reverification activities for a system shall be conducted by the verification team following the same basic procedures prescribed for the preceding verification stages.

4.2.3 Design Team

The composition of the design team shall be established in terms of the functions that are required within the team. The following functions are fulfilled by one or more people depending on project size and complexity.

4.2.3.1 Lead Engineer

This is the team leader, responsible for all technical matters in the development of the system. Normally one person is designated as the lead engineer for a project. The lead engineer shall have the responsibility for the development of the software design requirements and software design specification documents. Global decisions on the structure of the software, decomposition, and data base are made by the lead engineer. Some critical sections of the programs, both in terms of importance and complexity, may be coded by the lead engineer. The lead engineer supervises the rest of the design team in technical matters.

4.2.3.2 Programmer

A programmer's main responsibility is to develop the code and provide the details for the software design at the module level to meet the software design requirements. In most projects, it is anticipated that there will be more than one programmer.

4.2.3.3 Librarian

The maintenance of the software library is a key element in the V&V process. The librarian, in the execution of that position ensures that a project's software conforms to library standards, retains records of the software modules in use and revision levels, and assures the procedures for software changes are followed.

4.2.3.4 Language Expert

This team member supplies the technical information on the programming language that is used. This person is preferably one of the programmers.

4.2.3.5 Hardware Expert

The hardware expert's responsibility is to maintain all hardware in working order in the "as delivered" system configuration. The hardware expert should also have software experience in order to assist in writing software drivers. There could be more than one hardware expert per project.

4.2.3.6 Design Team Group Supervisor

The design team group supervisor is responsible for the technical adequacy of the work performed by design and requirements teams reporting to him/her. This supervisor also performs all the administrative functions for the system development in the project. These functions include the development planning, budget, and scheduling, as well as the necessary administrative interfacing.

4.2.4 Verification Team

The composition of the verification team shall be established by the functions carried out, similar to the manner of the design team. The following functions are fulfilled by one or more people depending on project scope and complexity.

4.2.4.1 Lead Verifier

The lead verifier is the team leader responsible for all technical matters concerning the verification of the system. The lead verifier has the responsibility for the development of the verification requirements and validation test procedures documents. It is also the responsibility of the lead verifier to check the documentation compiled by the design team for completeness and unambiguity.

4.2.4.2 Verifiers

The verifiers check the portions assigned to them with the use of the project validation test procedures and requirements documents. These checks are carried out by the verifier with any of the appropriate tools and techniques discussed that are agreed upon with the lead verifier. The verifier is also the independent reviewer for the design team. As is the case with the number of programmers in a project, it is anticipated there will be more than one verifier.

4.2.4.3 Verification Team Group Supervisor

The verification team group supervisor is responsible for the technical adequacy of the work performed by the verification team reporting to him/her. This supervisor also performs all the administrative functions for the system verification in the project. These functions include the verification planning, budget, and scheduling, as well as all the necessary administrative interfacing.

4.2.5 V&V Testing Strategy

4.2.5.1 Verification Team Responsibilities

Verification team shall evaluate the test documentation and results and supplemental testing as deemed necessary.

The emphasis shall be placed on assuring that the documentation detailing the system, hardware and software functional requirements and performance specifications are clear, accurate and complete.

The documentation shall be reviewed looking for omissions, inconsistencies, inaccuracies and errors of omission/irrelevant requirements. Some significant functional requirements may be identified and monitored as development progresses.

The test progress shall be monitored and the execution of tests and test results shall be evaluated. Supplemental testing shall be recommended when found necessary.

Full independent testing shall be performed without evaluating or using designers test data base/bed, or test results.

The emphasis shall be placed on full independent analysis of the system requirements and design specifications, independent testing and evaluation for the systems requiring the highest reliability.

The actual assignment of team members for engineering, verification, testing, and validation is shown in Exhibit 4-1.

Requirements and the implementation of design shall be evaluated to ensure that the resulting system operation is functionally correct and meets the performance objectives.

4.2.5.2 (Deleted)

4.2.6 Data Base Testing

It is not sufficient to test only the algorithm to verify the correctness of a safety related or non-safety related program. It is also necessary to establish the correctness of the data base used by that program. This potentially involves review of four different areas:

- data accuracy,
- data completeness,
- data structure, and
- data accessibility.

Data accuracy review deals with the correctness of the individual data items stored in the data base.

Data completeness ensures that all the data which needs to be present is in fact present in the data base.

Data structure review deals with the analysis of the structure of the data base. It may include the ordering of the individual data items within the data base as well as the structuring for accurate and efficient searches or access.

Data accessibility reviews determine the extent to which the data items could be modified, intentionally or unintentionally. Methods for "data hiding", that limit the ability to modify data to known software items, are preferred. These methods protect software against unintended function brought on by unexpected changes to data made by unauthorized program functions. In contrast, global data techniques that result in unrestricted access and modification are undesirable.

4.3 REQUIREMENTS PHASE V&V ACTIVITIES

4.3.1 Introduction

The intent of verifying the functional requirements is to ascertain that the requirements are complete, correct, consistent, clear, traceable, and testable. The selection of these evaluation parameters is based on their ability to assess the quality of the documented communication with the designer and also the user of the system.

The functional requirements form the basis of all the system design and verification efforts, and are used throughout the rest of the product life cycle. They serve as the basis for the verification of design specifications which, in turn, are the basis for the verification of design implementation. Functional Requirements are the bases against which all the validation activities are performed.

The principal purpose of a requirements document is:

1) To clearly define the objectives and needs of the system design & development process. Both the designer and the user must be able to understand and perform a meaningful assessment of the system.

2) To serve as a means against which an implementation can be validated and the intermediate steps can be verified.

The goal of verification activities during this phase is to ensure that the requirements documents do indeed serve the above purpose.

4.3.2 Formalization of Requirements:

In order to satisfy the need of both the user and designer to understand and evaluate the system, real-time system requirements are stated in terms that overlap the area of competence of application experts and the system architect/designers. Stating the requirements of a large real-time system explicitly and in easily verifiable mathematical form is currently a desired but possibly an unachievable goal due to the complexity of the requirements.

In practice, most of the requirements techniques exhibit a procedure or structure to show how certain output response of the system is generated in response to system stimuli and inputs. In any non-trivial real-world system, specifying all system requirements in explicit rigorous mathematical

statements, which are more readily testable, is likely to be impossible. However formulating the critical functions of a system in this way to the extent practical will be of significant value during validation of system level functions. The system requirements are usually categorized as a) functional requirements, b) performance requirements and c) special attributes with further sub-categories appropriate for the system decomposition.

As a common practice complex systems are systematically decomposed into smaller subsystems and their functions are assigned to either their hardware or software. In some systems in order to present a clear picture, decomposition may include data flow, control flow, and intricate synchronization and timing aspects and implicitly specify the software and hardware architectural requirements.

4.3.3 Practical Role of Verification Team:

Verifying the system architecture and decomposition is one of the V&V tasks. The interrelationship between hardware/software and subsystems are reviewed by the verification team to ensure that the overall integrated system does indeed have potential to meet the system needs and objectives. Thus in addition to system requirements, hardware and software requirements specifications are also verified for proper segmentation with appropriate and sufficient details to ensure system design and decomposition integrity.

4.3.4 Verification of Functional Requirements

1) The major objectives of the verification activities during this phase are to:

   a) Evaluate the adequacy of the allocation of system requirements to hardware, software, and subsystems.

   b) Evaluate the feasibility of accomplishing the system objectives and goals with the assigned requirements and using the allotted processor resources.

   c) Ensure design requirements are complete, accurate, testable, and unambiguous as possible.

   The tasks typically performed by the V&V team during system requirements verification are listed under Functional Requirements Verification Tasks.

2) The results of these activities produce a collection of approved requirements documents.

4.3.4.1 Expected System Design Requirements Errors

The following errors frequently occur.

1) Inadequate or partially missing performance criteria.

2) Inadequate or partially missing operating rules (or information).

3) Inadequate or partially missing ambient environment information.

4) Requirements that are incompatible with other requirements.

5) Inadequate or partially missing system mission information.

In addition to the above, other defects are: untestable requirements; erroneous external interface definitions; initialization of the system not considered; misstated or misinterpreted user needs; elements of flexibility and maintainability not specified.

4.3.4.2 Concepts and Methods Directing Functional Requirements Verification

Functional requirements verification activities shall be directed by evaluating the checklist in Exhibit 4-2.

4.3.4.3 Functional Requirements Verification Tasks

Identify and evaluate functional requirements of:

- Functional processing,
- Performance budgets,
- External Equipment Interfaces,
- Operational features,
- Man-machine interfaces, and
- Internal hardware and software interfaces

4.3.5 Verification of Hardware Design Requirements

The purpose of the hardware design requirements verification stage is to ascertain that the hardware design requirements are understandable, unambiguous, reasonable, implementable, accurate, complete, and are a faithful translation of the Functional Requirements into Hardware Design Requirements.

At the hardware design requirements verification stage the hardware in which the system will be implemented is generally not known. Therefore this stage concentrates on the feasibility of the design requirements in respect to the practical application of the system. The major concerns of the verifier during the hardware design requirements verification stage are items such as:

Are the required standards which are included up-to-date with the present technology?

Which government and industrial standards apply and are they up-to-date with respect to the technology and application?

Can the technology of today support the application in a reasonable and cost effective manner?

Do the design requirements include all required sections?

Do they follow standard documentation rules and guidelines?

4.3.6 Verification of Software Design Requirements

1) The software design requirements verification effort is initiated with the analysis of the software design requirements generated by the designer to ensure that the software will accomplish its allocated system requirements.

2) The software design requirements documentation is written so as to be easy to read, easy to understand, and difficult to misinterpret.

3) It must clearly specify the constraints imposed upon the software by the hardware and the environment.

4) A product of this activity is a set of authenticated requirements which shall become the allocated baseline for the Development Phase.

5) The software design requirement defines the software to be developed for the system implementation.

6) The requirements define the logical division and separation between software units, modules and subsystems.

7) The requirements shall also describe the program structure, the information flow, the logical processing steps, the data structures and their relationships.

8) The requirements include items such as performance, interfaces, timing and memory size constraints, support software and other hardware/software information pertinent to the program's operating environment.

4.3.6.1 Expected Software Design Requirements Errors

The errors that frequently occur during the software design requirements phase can be categorized as follows, and should be evaluated during verification:

- Inadequate or partially missing requirements for design to progress,
- Incorrect requirements,

- Requirements that are inconsistent or incompatible with other requirements,
- Ambiguous or misinterpretable requirements,
- User's needs not properly understood or reflected,
- Requirement not traceable to user's needs, and
- Untestable requirements.

In addition to the above, errors are observed in the following areas:

- Accuracy specified does not conform to the need,
- Data environment inadequately described,
- Input/output data parameters units incorrect,
- Erroneous external interface definition.
- Initialization of the system not properly considered,
- Vague requirements of the functions to be performed,
- Required processing inaccurate,
- Required processing inefficient,
- Required processing not necessary,
- Missing requirements on flexibility, maintainability,
- Missing or incomplete requirements of response to abnormal data or events,
- Inadequate or incorrect algorithm,
- Incorrect timing/synchronization requirements,
- Incorrect hardware interface requirements, and
- Incorrect allocation of system resources.

| 4.3.6.2 (Deleted)

4.4 UNIT & SUBSYSTEM DESIGN VERIFICATION

4.4.1 Introduction

The purpose of design specifications verification is to ascertain that the design specifications are a faithful translation of the design requirements before the design is committed for implementation.

The design specification documents define and provide the details of the system design structure, information flow, processing steps and other aspects required to be implemented, in order to satisfy the system design requirements. The intent of the design specification verification is to ensure that the design specifications are clear and understandable, accurate, correct, consistent, complete, implementable, testable, and traceable to the design requirements.

Considering the inherent iterative nature of design activities, V&V tasks are conducted on an ongoing basis. This is highly desirable especially when V&V efforts parallel design activities. Test planning and verifying the conformance of design documentation to established standards are the major objectives of preliminary V&V activities. As the design progresses, the design as documented is analyzed and critically evaluated for its potential to meet design requirements.

4.4.2 Verification of Hardware Design Specifications

The purpose of the hardware design specification verification stage is to determine that the hardware design specifications are understandable, unambiguous, reasonable, implementable, accurate, complete, and are a faithful translation of the hardware design requirements into hardware design specifications.

1) The hardware design specifications defines the equipment that will be developed or purchased during the implementation stage.

2) The specifications defines the logical division and separation between functions, modules, and subsystems.

3) Each unit is given a set of specifications which define the limits in which the hardware shall be designed.

4) These specifications include items such as performance, interfacing, electrical characteristics, modularity, compatibility, and operation.

In most cases the main processing hardware in which the design will be implemented is known. The specifications may specify that an industrial standard bus or processor and the like must be used in the design. In addition, new supporting hardware may be specified. This may include new printed circuit boards, signal conditioning boards, chassis, power supplies, or cabinets.

The major function of hardware design specification verification is to insure that a) the specifications translate properly and accurately the hardware design requirements, b) that the specifications are consistent, comprehensive, and complete, c) that all facets of the implementation have been defined and d) that clear and concise specifications have been written.

If the specifications specify that outside vendor hardware is to be supplied, the process of verifying vendor records and the gathering of verifying vendor information can proceed.

4.4.3 Verification of Software Design Descriptions

The Software Design Description is a detailed description of the software to be coded. It describes decomposition of the software into entities. Each entity is described by its type, purpose or function, subordinate entities, dependencies, interfaces, resources, processing and data.

4.4.4 Verification of the Integrated System Design Specifications

A verification of the integrated system design specifications is performed to ensure that the system design specifications properly reflect the system design requirements.

4.4.4.1 Objectives Directing the Integrated System Design Specifications Verification

1) Preliminary relationships are reviewed to ensure that module's and subsystem's functional and operational performances are appropriate.

2) The integrated systems design specifications are then reviewed for technical feasibility, testability, and maintainability related to the integrated system in its totality.

3) The tasks to accomplish the integrated system design specifications verification include the following:

- Evaluate the system functional features to determine if they accomplish the system design objectives.
- Analyze and evaluate the system operational characteristics to ensure that they conform to the system performance budgets.
- Evaluate the internal hardware/software interface.
- Evaluate the system man-machine interfaces.
- Evaluate the external equipment interface impact.

4.5 UNIT AND SUBSYSTEM IMPLEMENTATION V&V

4.5.1 Introduction

The purpose of the unit and subsystem implementation verification is to ascertain the implementation documents are clear, understandable, logically correct and a faithful translation of the design specifications. The objectives of the implementation documents are to facilitate the effective production, testing, use, transfer, conversion to a different environment, future modifications, and traceability to design specifications. In general the verification activities during this phase are oriented towards evaluating the following:

1) Does the implementation satisfy design specifications?

2) Does the implementation follow established design standards?

3) Does the implementation follow established documentation standards?

4) Does the implementation serve production, test, use, transfer and other needs that motivated its creation?

5) What is involved in testing the actual resulting product?

The unit and subsystem implementation verification activities also include verification by inspection a) of source code program or b) drawings, as appropriate, to assess the performance of either derived or resulting assembly. Based on the results of evaluation and analysis of the implementation specification, also during this phase, the verification team reviews detailed test procedures in preparation for the next stage of implementation testing. Documentation of this review shall consist of completing the appropriate sections of Checklist #3 (Exhibit 4-4).

The primary implementation verification method is systematic evaluation and analysis using checklists. These checklists contain an extensive list of questions that are utilized to evaluate the implementation and its documentation.

4.5.2 Verification of Hardware Implementation Specification

4.5.2.1 Introduction

The purpose of the hardware implementation specification verification stage is to ensure that the hardware implementation specifications are understandable, unambiguous, reasonable, accurate, complete, and are a faithful translation of hardware design specifications into hardware implementation documentation and the generation of hardware verification test procedures.

Inspection is the primary method used in the verification of implementation specifications. Inspection also confirms that drawings are accurate and complete.

4.5.2.2 Hardware Implementation Verification Considerations

There are two general categories of hardware: electrical assemblies, and mechanical assemblies.

Electrical assemblies can be printed circuit boards of any size or function such as mother and daughter boards, test panels, chassis, connector panels, termination strips (with electrical components), relays, switches, power distribution centers, power supplies, fuse panels, indicators, blowers, buses, and special purpose chassis. Electrical assemblies are usually a combination of mechanical and electrical assemblies such as a computer chassis (hardware).

Mechanical assemblies are cabinets, chassis, bus ducts, cable trays, racks, cables, connectors, supports, fasteners, grounding bars, pumps, breakers, fittings, as well as many others. Mechanical assemblies do not necessarily have to be a part of electrical assemblies.

The verification requirements for mechanical assemblies are significantly different from electrical verification requirements.

1) The primary method for mechanical assemblies shall be inspection.

2) Typically the mechanical assembly shall be inspected for items such as paint uniformity, color and thickness, physical dimensions, proper materials and correct assembly.

3) The basis for inspection shall be the implementation specifications.

4) The verification process shall insure that the proper materials, coatings and assembly instructions were specified.

5) The verification method used for electrical implementation specifications shall be inspection. The implementation documents shall be verified for proper content and format.

6) The drawings shall be checked to insure that the implementation properly reflects the design specifications.

4.5.3 Verification of Software Implementation

1) Implied in the definition of verification is the assumption that there exists documentation that specifies completely and unambiguously how the software will operate.

2) Verification of the implementation specifications is started after the development team confirms to its own satisfaction that the software is correct and operational.

3) At this point an error incidence report is formally established.

4) All errors during verification are documented in a Test Exception Report (Exhibit 8-1). Corrective action is taken according to problem reporting and corrective action, Section 8.

5) The programming language used is as described in the software design requirements.

6) Testing is the method for object program verification.

7) For source program verification, inspection is used and for validation, testing methods shall be applied.

8) Testing oriented V&V activities is performed according to Section 4.6 and 4.7.

9) Inspection is the verification method that is applied, during this stage, to the source program as part of the software design implementation verification.

4.6 TESTING PROCESS

4.6.1 Introduction

This section details the testing process used as part of the V&V plan. It defines a set of requirements, necessary for the generation and utilization of verification test procedures. These requirements cover the areas of test features selection, test case design, test development, and processing of test data. This section includes hardware, software and system testing considerations.

This section does not address or identify particular test techniques or tools that are used to accomplish the specified tasks.

The testing process shall comprise the following steps:

- SELECT features to be tested,
- PLAN the approach, resources, and schedule,
- DESIGN the set of tests,
- IMPLEMENT the plan and design,
- EXECUTE the test procedures,
- CHECK resulting conditions, and
- EVALUATE the results

The verification test results document is comprised of the final results of the testing activities.

The verification test procedures documents are linked to the verification test results documents and to the other documents of the total V&V activities.

The overall set of documents covering definition, implementation and reporting of all test activities is sometimes referred to as a test program.

Since the definition and implementation of the testing process is one of the most important aspects in the overall Verification and Validation process, Checklists #1 - #7 are used to insure that all aspects have been considered. (Exhibit 4-2 through 4-8.)

4.6.2 Hardware Testing Process

4.6.2.1 Selection and Categorization of Hardware

1) The following basic questions are answered before selecting hardware for testing. These questions are:

- What requirements exist that determine the level of testing required?
- Who designed the hardware? A vendor or is the design internal?
- How much of the hardware has been tested prior to this project?
- If the hardware was supplied by a vendor - can the vendor show proof of design control?

2) Once the above questions have been answered the next step is to categorize each piece of hardware.

- Category one: Vendor supplied hardware.
- Category two: Hardware that will not require testing.
- Category three: Internally designed hardware.

Category one: Vendor supplied hardware requires the verification that the design and performance of the hardware meets specifications. The process of determining that the hardware is satisfactory is similar to category three except that the testing will have been done by the vendor.

Category two: Hardware that falls into this category has been previously tested and proven in another product. This hardware usually must meet the same requirements as do categories one and three.

Category three: Hardware in this category usually must be tested. This hardware can be either internally designed or vendor hardware that did not meet the necessary requirements. Also, hardware in this category may not be required to be fully tested. It may be vendor hardware that needs only one test to complete the vendor testing

requirements or a prior product that only needs some special testing for it to be verified.

3) A list is developed that categorizes the hardware, and includes what verification rules apply.

4) The functional requirements, system description, hardware design specifications and hardware implementation specifications and vendor supplied specifications are the primary sources for testing information.

Each piece of hardware shall be evaluated and a determination made as to the items that are to be tested.

4.6.2.2 Hardware Verification Test Planning

Once the hardware that is to be tested has been identified, the process of planning the design of the tests can proceed.

1) The hardware is tested from the simplest module on upward to the most complex modules.

2) The tests are designed as to support the simplest as well as the most complex.

In this way the cost of testing as well as the time required is reduced. Consider the following approach:

- Arrange the hardware in the order of complexity.
- Organize the hardware into families.
- Make a list of features and items to be tested.
- Select the tests to be developed based upon the common features and items. For example memory tests, I/O tests, and the like.
- Select the rules that will govern the development of tests.

3) Using the above list, a comprehensive list of common features is developed.

4) Test details such as what each test must perform and the options desired in each test is developed.

5) A set of test specifications is developed that details what test shall be developed and the functions they perform and the hardware they test.

4.6.2.3 Hardware Test Design Considerations

Hardware modules and the items and features to be tested shall be listed along with the identification of each module including its revision level and version.

4.6.2.4 Implementation of Hardware Tests

1) The implementation comes in the form of a set of step-by-step instructions that direct the testing and shall be part of the hardware validation test procedure (HVTP).

2) Each test procedure will be thoroughly documented to allow an independent party to perform the test.

3) An HVTP shall be comprised of two parts: the first part includes general information such as the identification of the equipment to be tested, the identification of test equipment that will be used for the tests, test matrix, and revision control sheets. The second part includes the individual test procedures (TP) for the hardware under test.

4) The following lists the information that is included in a test procedure:

- Test equipment: the test procedure includes all test equipment that is required to perform the test(s). This includes everything from a simple voltmeter to a complex computer system that is being used to test the hardware. Where applicable the calibration of the equipment must be up-to-date. Where necessary the equipment should be tested before each test to insure that the test equipment is functioning properly, that the hardware under test will be tested properly, and that the results will be valid. The serial numbers and any identification that is possible shall be included in the test procedure.

- The test set-up: The test set-up includes all necessary hook-ups, set-ups, cabling, loading instructions, voltages, tools, instruments,and any instructions that may be necessary for a technician to perform the test(s). A drawing is the preferred method for showing the test set-up.

- The test procedure (TP): A TP shall be a step-by-step procedure that gives the exact sequence to perform a test. A series of TP's shall comprise a HVTP. A TP should be written such that a technician can perform the test without the aid of a verification engineer. The TP shall be detailed in such a manner that there are no questions as to what must be done. All inputs, values, sequences, keystrokes, outputs, tools and any other instructions shall be included. The instructions shall be clear and concise. A collection of TP's form the complete testing of a piece of hardware and therefore the TP's should be arranged in such a way that the simplest tests are performed first and each succeeding TP encompasses a more significant aspect of the hardware.

4.6.2.5 Execution of Hardware Verification Tests

1) Once the tests have started any errors other than the initial set-up problems are recorded.

2) A design error or any questions concerning the design and its operation is resolved by the designer.

3) Once the error has been resolved the tests are re-run starting from the point that the verification engineer instructs.

4) All events that occur during testing are recorded and these records maintained.

4.6.2.6 Verifying the Results of Hardware Verification Testing

1) The results of each test are checked in detail to ensure that the test was properly executed and that the results meet the expected results.

2) Each printout, measurement, indication and any other way of determining the correct results are checked and if necessary a proof given so that the results can be understood.

3) If there is any question as to the results of a test, the test is re-run.

4) If a test fails, the verifier generates a Test Exception Report (Exhibit 8-1).

5) The designer finds the reason for the failure and corrective action taken according to Section 8.

6) The failure and the corrective action is recorded by the designer.

7) Once the reasons of the failure have been discovered, the appropriate corrective action paper is completed and sent to the responsible party for corrective action.

4.6.2.7 Evaluation & Generation of Hardware Tests Results Report

The final step is similar to the check step.

1) The results of the testing process are reviewed and comments and result of each test is given a pass or fail mark.

2) If the conclusion is to pass, then the hardware under test shall have completed the verification process.

3) If the conclusion is that the hardware failed, then the hardware is evaluated to determine what must be re-designed or modified.

4) If the hardware failed, it is re-tested after the appropriate action has been taken.

4.6.3 Software Testing Process

4.6.3.1 Selection and Categorization of Software

Starting with the Functional Requirements, Software Requirements document and Software Design Description, a list which includes the following shall be created:

1) Name: Identify the test items by name and include version and revision level.

2) Category: Categorize the items to be tested (software unit, module, and subsystem).

3) Feature: Summarize the items to be tested, including features to be tested, and interfaces with other items.

4.6.3.2 Software Validation Test Planning

Using the list of test items and their features to be tested, describe the overall approach to testing which will ensure that these items and features will be adequately tested. Specify the major activities, techniques and tools which will be used.

The Software Validation Test Plan shall be independently reviewed. Documentation of this review shall consist of completing the appropriate sections of Checklist #4 (Exhibit 4-5).

4.6.3.2.1 Approach to Software Testing

The type of testing to be done is identified in terms of:

- Areas to be covered (i.e. functions, instructions, etc.) and degree of coverage required,
- Data value selection rules (e.g. boundary-value, typical value, etc.) to be used, and
- Techniques to be used for output recording, collection, and analysis.

4.6.3.2.2 Resource Requirements

The test plan is organized in such a manner as to group all items with the same features.

Consider the test environment required in terms of hardware, test tools, test team organization, and other items that must be present.

4.6.3.3 Implementation of Software Tests

1) Starting from the information derived during the test design, and input test bed, expected output test results shall be prepared. The procedure to run the test data is set up in terms of:

   a. Sequence of actions necessary to begin test execution,

   b. Sequence of actions necessary during execution of test run,

   c. Sequence of actions necessary to suspend testing, when unscheduled events dictate,

   d. Sequence of actions necessary to restart the test run after unscheduled events,

   e. Sequence of actions to halt the test run,

   f. Method for building, updating, and maintaining the test data base, and

   g. Environment used for the test execution.

4.6.3.4 Execution of Software Validation Tests

1) After setting up the test environment as called in the test procedure (TP), the test shall be conducted by executing the step-by-step instructions of the TP.

2) A chronological record of the test execution is prepared and the pertinent results and actions information are recorded in a Test Log.

3) The test log shall contain:

   a. Test-log identifier, and
   b. Description.

Identifier: identify the items being tested including their version levels, and the attributes of the environments in which the testing is conducted. Include facility identification, hardware being used, system software used and resources available.

Execution description: for each execution, record the visually observable results (for example error messages), input and output data, personnel responsible for test execution, record circumstances surrounding any anomalous event.

4.6.3.5 Verifying the Results of Software Validation Testing

1) Starting from the execution information recorded in the Test Log, a check for test failures is made utilizing the expected output test results.

2) Based on the type of fault, as determined by the failure analysis, appropriate corrective action are taken and the impacted test cases are re-executed.

3) The design failure notification report shall be in a department standard format or as specified in CM plan.

4.6.3.6 Evaluation and Generation of Software Tests Results Report

1) The software design and implementation is evaluated based on the test results.

2) If the conclusion is that the software failed the validation testing stage, then the developer shall evaluate the software to determine what shall be re-designed or modified.

3) Re-validation shall be necessary after the appropriate action has been taken, according to the revalidation after modification practice.

4) If the conclusion is that the software passed the validation testing stage, then it shall be released.

5) In both cases (pass or fail) test results shall be maintained in accordance with configuration management plan.

6) Software test results shall be independently reviewed. Documentation of the review shall consist of completing the appropriate section of Checklist #5, Exhibit 4-6.

4.6.4 System Testing Process

1) It is the practice that new system designs are verified to ensure that the system equipment meets the functional requirements and specifications.

2) In order to minimize the number of test cases required for a total integrated system test, a sufficient number and type of prerequisite tests are conducted.

3) For the series of tests described by the outline referenced in Exhibit 4-1, the following activities shall have taken place, and thus credit shall be taken for them when designing the tests:

   a. All hardware modules shall be verified to operate in accordance with their Hardware Design Specifications.

   b. All software modules and processes shall be verified to operate in accordance with their Software Design Descriptions.

   c. The system shall be functionally checked to ensure that the signal paths are in accordance with design documentation.

4) The final set of tests are called the integrated system test and shall be conducted with all system elements connected together.

5) It is the standard practice that test cases are defined for each step of the system validation testing which ensures that the equipment being tested is exercised over its normal and abnormal range of inputs in a way that assures its proper operation in an actual plant situation.

6) The test cases specified are appropriate to the intent of the test and are selected with the knowledge of prior test runs. In particular, it is intended that where a series of overlapping tests is defined, that the test cases selected for the latter test make use of the knowledge gained from the earlier tests.

7) Test cases are defined with consideration being given to the internal and external architectural features of the system. For instance, if a circuit board or software module is used in several applications, a rigorous test of only one board or module is required, while for the others, only a limited test, primarily to check interfaces, is made.

This approach will also be used where multi-variable inputs to a portion of the system occur. In general, the test cases are chosen so that one variable is exercised while others are held at a nominal value related to normal operating conditions.

8) For complex circuits (e.g., the DNBR and KW/FT, and the trip logic modules in protection systems), additional test cases are selected to ensure that the complex interactions within the circuits function properly.

9) The input signals used for the system validation test are simulated. The simulations are such as to reproduce the type of signal produced by the actual sensor in a plant. Process noise or any other noise shall not be simulated as this will obscure the information that is being collected; more specifically, that the overall transfer function is in accordance with the system design specification.

10) The two types of inputs used to exercise the system shall be steady state and dynamic. Both types shall be used depending on the nature of the information to be collected. For instance, channels with no time dependent elements shall be adequately checked using slowly varying or incremented signals to establish gain and bistable accuracies whereas channels containing lead lag units, etc., are verified through the use of time varying signals whose dynamic characteristics shall be precisely known. Dynamic input signals consist of ramps and steps (the latter being a special case of the former). The rate of change and magnitude of these signals is chosen to be consistent with the response of the channels being tested. Steps shall be typically used to establish the time response of a channel and its transient response generally while ramps are used to demonstrate the correct system operation and signal polarity.

11) Tests are run that demonstrate the proper operation of the system in response to out of range conditions. These tests are designed to drive the channels into saturation to ensure that the system responds in an orderly and timely manner when the out of range condition is removed.

4.7 UNIT AND SUBSYSTEM VALIDATION TESTING

4.7.1 Introduction

The unit and subsystem validation testing stage is the last and most important stage for a unit or subsystem. In this stage the hardware and software products of all previous stages converge into tangible hardware and software. The hardware and software now go through the most rigorous stage of all, the testing stage. The hardware and software usually will be tested for every aspect of its design.

At the beginning of this stage all documents concerning the products to be tested have been verified for completeness and correctness. The documents that have been verified are the Functional Requirements, System Description, Software Overview, Hardware and Software design Specifications, Hardware Implementation and Software Implementation Documentation.

Usually all test procedures have been written that will direct the testing. These procedures are a step-by-step process of validating the design. This stage is primarily the execution of the test procedures and the reporting of testing results.

4.7.2 Execution of Hardware Unit Validation Tests

Unit validation testing occurs in four stages: the development of test tools and systems, the writing of detailed test procedures, the execution of the test procedures, and the analysis of test data.

4.7.2.1 Hardware Unit Validation Tools and Systems

1) The development of test tools and systems are started on or before the hardware design specifications are released.

2) At the beginning of this stage the test tools are complete or receiving their final touch-ups.

3) Validation test tools and systems are designed to make the validation testing as simple as possible, as automated as possible, and as accurate as possible.

4) The test tools encompass all aspects of the hardware that is to be tested.

5) Unit testing is based upon a bottom-up approach. In bottom-up testing the simplest modules shall be tested first.

6) Modules are selected so that each module tested supports the next highest level of complexity.

7) Typically a module is tested for all functions, operations, voltage margins, temperatures, temperatures at voltage margins, inputs, outputs, accuracy, and stability.

8) Bench testing is used to determine power consumption, internal voltages, clock rates, precision sources and simple basic operations.

9) Where applicable system type testing is used to test a module.

10) Each part of the implementation specifications and design specifications is tested to determine if the hardware satisfies these specifications.

11) The intensity in which the units are tested vary depending on the application and other requirements that govern the verification process.

4.7.3 Execution of Hardware Subsystem Validation Tests

1) At the unit level, single modules and functions are tested.

2) At the subsystem level these units are combined and tested as a subsystem.

3) At the subsystem level the bus is fully loaded and at its maximum length.

4) Unit interactions are tested in accordance with the test plan.

5) The units at the subsystem level have some stimulus applied so that the stability and accuracy of the module can be measured with respect to the total subsystem stability.

6) Subsystem validation testing occurs in four steps: the development of test tools and systems, the writing of detailed test procedures, the execution of the test procedures, and the analysis of test data.

7) Subsystem testing is based upon a bottom-up approach. In bottom-up testing the simplest units and functions shall be tested first.

8) Tests are selected so that each module tested supports the next highest level of complexity.

9) Typically a subsystem is tested for all functions, operations, voltage margins, temperatures, temperatures at voltage margins, inputs, outputs, accuracy, and stability.

10) Each part of the implementation specifications and design specifications are tested to determine if the hardware subsystem satisfies these specifications. The intensity in which the modules are tested shall vary depending on the application and other requirements that govern the verification process.

4.7.4 Preparation of Hardware Test Results Report

1) The hardware validation test results report is divided into two parts.

2) The first part contains all the information necessary to perform the tests and record their results.

3) The second part contains the analysis and conclusions of the test data. The conclusions are to accept or reject the hardware.

4) The hardware validation test report contains the following general information: title page, information concerning use of document, record of changes, list of effective sections, table of contents, list of tables, list of figures, purpose of tests, identification of the hardware being tested, test details, test procedure verses the test matrix, the test procedures, and the results of the testing.

5) The hardware validation test procedures are contained within the hardware test report.

6) The hardware validation test procedures contain the following information:

- The test equipment to be used, a description of the test equipment, the model and serial numbers of all test equipment to be used, the manufacturer of the test equipment and the calibration due date,

- The test set-ups which contain all the required information necessary to prepare the hardware for testing. Items such as cabling, power supply hook-ups, and test equipment hook-ups, are in sketch form and are accompanied with all necessary instructions and check-out procedures. The test set-ups give a sufficient amount of information so that an independent party can perform the tests,

- A data record sheet on which all recordings, printouts, recorded values and the like are to be recorded, and

- The step-by-step instructions on how to perform the tests, the readings to be taken, the input values, and any other needed information are contained in the validation test procedure.

7) Each test and the recorded information shall be signed-off by the person performing the tests.

4.7.4.1 Hardware Verification Test Procedure Conclusions

1) When all testing is complete on a particular set of modules the conclusions part of the test report is prepared.

2) The formation of a conclusion is a step-by-step analysis of all the recorded information.

3) Each recorded item is checked for the proper values, responses, limits, and any other parameter that is applicable, and the results shall be compared to the implementation specifications.

4) Each item in the implementation specification is verified as being acceptable or unacceptable.

5) The final conclusion is either pass or fail.

4.7.5 Software Unit Validation Tests

4.7.5.1 Introduction

1) At the unit level the software is not integrated into the system.

2) The unit is isolated from the rest of the system so it can be extensively tested, normally in a host environment first and successively in a target environment.

3) Each unit is tested in order to have the maximum topological coverage (all instructions, all branches, all paths) compatible with the criteria described in the test plan. Results are checked with precalculated values.

4.7.5.2 Expected Software Unit Level Errors

Unit testing includes two major areas, logic testing and computation testing. The purpose of logic testing is to find and correct errors in the code and resolve potential abnormal behavior, such as:

1) Stopping or trapping executions in a loop,

2) Incorrect logic decisions,

3) Lack of logic to handle certain legitimate combinations of input conditions, and

4) Lack of logic to handle missing input data.

The logic should be exercised to determine that only the expected outcome is obtained.

The purpose of computation testing is to find and correct errors in the computation sequence, performance, accuracy, timing and proper initialization of numerical algorithms for:

1) Data within specifications,

2) Data outside specifications (abnormal data),

3) Boundary data,

4) Singularity data, and

5) Fixed point and units.

4.7.6 Software Subsystem Validation Tests

4.7.6.1 Introduction

At the subsystem level the software is integrated into the subsystem hardware. The purpose of subsystem level test is to show the proper integration of the software modules and that functionally related modules correctly interact to perform the intended function. It involves additional physical resources over and above that required for the single module testing.

4.7.6.2 Expected Software Subsystem Level Errors

1) Subsystem testing shall address the following problems that may be expected to occur at the subsystem level:

   Linkage between modules such as:

   - Data and control flow,
   - Order, type and number of arguments in subroutine calls, and
   - Absence of spurious outputs from, or undesired changes in, a global data base.

2) Timing behavior,

3) Intended sequence of events including:

   - Setting or resetting global flags, and
   - Initialization of variables,

4) And possible undesirable behavior of feedback loops

4.7.7 Execution of Software Unit and Subsystem Validation Tests

The Execution of test cases includes:

1) conducting the test according to test procedure,

2) preparing records required by test procedure, and

3) taking actions required by the test procedure.

4.7.7.1 Software Test Equipment

1) The software is tested with a collection of tools applied with engineering judgment, both in the host and target environments.

2) The application of one tool instead of another will depend upon the particular test activity that shall be carried out on a particular module.

3) Testing in the host environment shall be essentially structural testing.

4) Testing in the target environment shall be essentially functional testing.

5) Software drivers, together with structural testing supporting tools provide the major support to test software that is resident on the host computer.

6) The software is exercised efficiently and extensively.

7) Statistics are gathered and the results shall be presented in an organized manner that is readily reviewable.

8) In-Circuit Emulation of the microprocessor CPU is a powerful tool for testing in the target environment and is used as required.

9) Partial and conditional execution of the software is also possible.

10) Simulation of external signals is used in software testing in the target environment.

By simulating external signals the software will execute in an environment that closely approximates actual operating conditions.

4.7.7.2 Software Test Application

1) Tests are applied to the software and documented by the design team or V&V team according to Exhibit 4-1.

2) Single modules or units which do not interact directly with the hardware shall be tested on the host computer either by structural testing or by functional testing in relation to the modules (group of modules) application to safety.

3) For functional testing of single modules or groups of modules the external environment is simulated by a test program or a software driver, which shall run on the host environment.

4) Results of functional testing are compared with precalculated values which may be obtained either from the Software Design Requirements or by off-line computer modeling.

5) For structural testing, single modules or a group of modules are tested with test data derived either from an analysis of the code or from the program specification.

6) A testing coverage tool, where possible, is used to keep track of the percentage of instructions, branches and paths covered by the test cases.

7) Modules or subsystems which interact directly with the hardware are tested on a Microprocessor development System with an In-Circuit Emulator, as required, which enables the tester to interact with the software on the actual hardware.

8) External signal simulation is used whenever it is judged necessary by the chief verifier.

4.7.7.3 Software Test Inputs

1) In general most of the test inputs at the unit and the sub-system level are supplied by the design team and reviewed by the V&V team.

2) Although safety criteria may require some tests to be performed (especially at the system validation level), and the design team may recommend some tests or test strategies, it is the V&V team who is responsible for the correctness of the software and for the actual test selection and test application.

3) Testing techniques are divided into different categories.

4) Usually no single technique will be sufficient for testing a module, and so a combination of them are used.

4.7.8 Preparation of Software Test Results Report (STRR)

1) The Software Test Results Report provides the complete test results.

2) In conjunction with the Software Validation Test Procedures and the Software Verification Requirements, the STRR contains enough information to enable an independent party to repeat the test and understand it.

3) The Software Test Results Report includes:

a. Input Test Listing: This is a list of the actual test inputs applied to the software.

b. Output Test Listing: A list of the actual test outputs.

c. Additional Data: This section gives all the additional data needed to repeat the tests. It shall include timing information, sequence of events, etc.

d. Cross Reference: This section refers to the corresponding sections in the Software Validation Test Procedure.

e. Conformance with Criteria: This section summarizes the test results in a narrative format. It refers to the test criteria in the Software Validation Test Procedure and determines if the software does or does not conform with it.

f. Error Incidence Report: This section is a log of all the errors discovered during the Validation phase. It includes the relevant information on the error, its type, the author of the defective module, and the cause of the error. It is used to detect weak links in the software generation chain.

4) The test report shall document every step taken during the test and the test results. This description shall be complete enough to repeat the process and thoroughly correlate new results with original results.

4.8 SYSTEM VALIDATION TESTING

4.8.1 Introduction

The verification process has provided an orderly step-by-step assurance of a true translation through the requirements, design and implementation phases, each step being assessed upon the basis of the previous step. The integrated system validation process involves determining whether the system meets it's functional requirements e.g., functional operations, system level performance, external interfaces, internal interfaces, testability, and other requirements as stated during the definition phase. System validation evaluates the system performance in an environment which is real, or as close to real as can reasonably be created; therefore, the fully integrated system with the actual system hardware and software is required. In large system applications, it may be required that validation testing begin at the subsystem level. Subsystem validation is usually desirable, to ease the error/failure isolation, even if not mandated.

The validation test environment must be configured to fit the system being tested, it should be matched to the available resources as much as practical to create the real operating environment.

The system validation process usually involves operational exercises of the system to assure the functional requirements are met. In some instances, system validation activities overlap those conducted earlier during verification and/or subsystem validation. Typical validation tasks are listed below:

1). The system functional operations are validated using the "black box" method, i.e. validating the system outputs by means of actuating prescribed inputs. Validation is conducted using the limits and ranges as designated in the system functional requirements, which are included in the system design requirements. The major validation areas shall be:

- Functional operation,
- System level performance,
- External interfaces, and
- Internal interfaces.

2) Abnormal conditions to be accommodated by the software shall be tested to assure that resulting functional operations are as described by the software design requirements.

3) Tests are executed to validate system functional operations during input transient conditions.

4) Independent validation tests and scenarios as required to validate the system design.

5) System validation procedures are updated, if required.

6) Final developer's documentation is evaluated to be:

- Complete,
- Accurate/compatible with delivered system, and
- Compliant with standards.

7) Validation test results are evaluated to be:

- Complete/consistent with procedures,
- Traceable to functional requirements, and
- Document results in Test Results Manual.

4.8.2 Expected System Errors

The following are possible system errors:

- The inability of the system to accomplish specific system functions.
- The inability of the system to respond to multi-variable transients such as those the system would experience while performing its mission in the plant.
- The inability of the system to satisfy system response time for each actuation function.

4.8.2.1 Concepts and Methods Directing System Validation

1) System validation is conducted to ascertain that all the requirements of the system are fulfilled.

2) The following problems are addressed by all parties that participate in system validation:

- Creating an effective set of test cases,
- Creating an efficient test environment,

- Managing test data, and
- Evaluating when to stop testing.

3) Validation test preparation steps, developed to address the above problems, include:

- Designing test cases against established test criteria. Test criteria for designing test cases for a specific system will be established in the System Validation Requirements Document.

- Creating a test database that relates each test to its requirements and manages test cases and test results.

- Setting pre-defined, realistic goals against which test accomplishment can be measured.

4.8.2.2 Examples of Test Goals

1) All the system operational functions, functions that must be accomplished by the system are tested individually in normal and abnormal operations.

2) Combination of functions are tested,

3) System tolerance to failure is tested.

4) Built-in test capabilities are tested.

5) Time response to defined events are tested.

4.8.2.3 Effectiveness of Simulation as a Validation Tool

Time itself is the essence in real-time systems and no amount of individual module testing can validate the temporal performance of the complete system. Only the real world or, more likely, an accurate model of the real world can provide realistic input test data for the final stages of system validation.

The use of a simulator for system validation is the nearest one can get to real life. However it should not be thought that this approach makes trivial real-time system validation. Accurate simulation can be very expensive and real-time simulation even more so. Precision simulators for

a large process plant can take many man-years to construct. For this reason, the use of simulation techniques must be carefully considered in relation to other validation techniques. Not least, whenever there is a discrepancy between real behavior and expected behavior, there is still the problem of determining whether it is the system under evaluation or the simulator itself which is at fault. It is most important that any such problem revealed by this procedure is carefully analyzed before a correction is applied: one must not correct the real world to fit the error. However it is often the case that this identification involves much work in analyzing the plant or system requirements specification. Because of the complexity of the equations which model the real plant and have to be solved at very frequent intervals, it is not uncommon to find that the simulation does not perform in real time.

However for validation, a simulator has a great advantage in that while both the system specification and the simulator may be incorrect because of difficulties in understanding the real plant, they are not usually both based on the same development personnel or environment. Consequently, any inconsistencies are less likely to coincide and testing with respect to a simulator may, in suitable circumstances, offer enhanced reliability expectations. It is especially noteworthy that the requirements specifications for the simulator and control system are likely to be separately derived. This independence between control system program and plant simulator is enhanced by the fact that simulators are often constructed from standardized modules and can to a large extent, be validated independently from any specific application.

The use of a simulator in system validation can, on its own, only lead to a confidence in the system of the same magnitude as the confidence in the accuracy of the simulator itself. Like testing techniques, the use of a simulator only detects anomalies; no anomalies detected does not prove that the system under investigation is error-free. For safety-related systems, much caution is needed and a simulator should have been extensively used before it is accepted as part of the overall validation procedure.

Despite the controversy of this section, simulators can be important in real time system validation. When the process plant to be controlled is too dangerous to be used for testing the system, or the plant is not yet in existence, a simulator provides the only means of checking the system in terms of its real world requirement rather than as a computer system on its own.

4.8.3 Execution of System Validation Test

1) Integrated system validation test is conducted in accordance with the Integrated System Validation Test Procedures. The inputs for each test case is produced by:

   - A Real-time simulator capable of driving the integrated system in real-time.
   - Simulating plant transients on a digital computer, using safety analysis codes, recording those transients on an analog tape recorder and injecting them into the system prototype through its normal inputs.
   - Manually

2) As the simulated transients are injected into the integrated system prototype, the response of the system is measured by recording various actuation logic signals.

3) These signals include system level actuation output, such as the reactor trip breaker actuation signal and the system level safety injection actuation in a protection system, and certain internal signals, such as the individual partial trip signals that are the inputs to the trip logic module.

4) The acceptance of the test results is based on a comparison of the actual measured first occurrence of the logic signals transitions to the predicted response of the system. The minimum set of output signals to be recorded for each simulated transient used as a test case must be selected.

5) The logic actuation expected to occur within the established time response.

4.8.4 Preparation of System Validation Test Report

1) The verifier is responsible for producing the system validation test report.

2) This document and the system environmental validation test results, are combined to provide the system test results manual. The system

validation test report should include the following minimal information:

a. Title page including all pertinent front matter information.

b. Introduction including scope and objectives,

c. Identification of what modules, subsystems, or system functions are to be validated.

d. Explanation of tests and test results. This explanation shall be in the form of a data sheet, and it should be self-explanatory regarding testing the system (e.g., by tag number, serial number, and/or rack/cabinet numbers). The data sheet should also identify the input signals, with the desired and actual outputs. A space should be provided for required accuracies and the test results. Space shall also be provided for the tester's signature and the acceptance sign-off by another engineer or verifier. Graphic data obtained during testing, such as recorder charts, computer printouts, etc., shall be used as additional documentation.

e. Summary of the test results and error incident reports.

f. Any revisions made to the system that would require additional validation tests and document changing shall be maintained through configuration management control.

4.9 OPERATION AND MAINTENANCE V&V

4.9.1 Introduction

1) Change controls apply to software released for production or repeated use.

2) Change controls are applied to all items in the library including those with prototype status. The maintenance cycle is entered after the system product has been released for production.

3) Verification activities shall include

   a. interfacing with the Library,

   b. compliance with change control implementation procedures,

   c. ensuring dependable code generation and loading,

   d. evaluating code/data changes impact on system execution and operation, and

   e. revalidating a production system to ensure that changes have not caused new problems.

4) Design changes and possibly revalidation testing shall be required due to the following:

   a. Errors not discovered during original design/testing.

   b. Failures in the entities external to the system, that can be corrected by changes to the developed system.

   c. Changing requirements for use of the system.

   d. Enhancements to improve efficiency, or to compensate for deficiency etc.

   e. Plant or application specific input/output data composition changes.

4.9.2 Modification Practice

1) The procedures established in the Software Configuration Management Plan are followed when considering a modification.

2) The following items are examined when considering a modification:

- Technical feasibility,
- Impact upon hardware and/or any other equipment,
- Impact upon software including a list of affected modules,
- Impact upon performance (i.e., speed, accuracy etc.),
- Necessary effort for reverification, and
- The documentation required for a modification.

3) Field modifications are performed only according to a strictly controlled procedure. The modification procedure depends upon the level at which it is introduced; design requirements, design specification, or implementation.

4) After implementation of the modification, a reverification process shall be conducted according to established procedures as mentioned in Section 4.9.3, Reverification After Modification.

4.9.3 Reverification After Modification

1) It is the responsibility of the Chief Verifier to decide case by case the extension to which to execute the new verification process.

2) The analysis of the reverification needed is documented in an auditable form.

3) As a general rule:

a. For a change at the requirements level the whole development process for any part of the system impacted by the change is re-examined.

b. A change at the subsystem design level is reviewed in terms of its potential impact upon corresponding lower levels.

c. A change at the module implementation level will require retest of:

- The affected module,
- The affected module interface,
- Hardware/software interfaces affected by the module, and
- Those portions of the total system affected by the change.

4.9.4 Change Controls

1) This communication is formalized by utilizing standard forms such as Request for Engineering Action or others as specified in the Software Configuration Management Plan.

2) The design team's responsibility is to evaluate the request for change and to take appropriate action including implementing design changes if warranted.

3) The designer is responsible for documenting a) the nature of the problem leading to the change, b) the nature of the change, c) any analysis required to understanding the change, d) other impacted items, and e) notification to all concerned and impacted parties, including the V&V team.

4) The role of the V&V team is to evaluate the implementation of the design changes and to take appropriate action, including the conducting of verification and validation tests, as needed to ensure the integrity of the affected system product. The administrative details for the change control mechanisms are provided in the CM Plan.

4.9.5 Software Library

1) In the development phase, the source code shall be written, compiled and tested by the development team.

2) If necessary it is modified, recompiled and retested.

3) When the design team is satisfied with its performance, it is the responsibility of the chief programmer to release the module or subsystem source code requesting entry into the library.

4) A software item is entered into the library only after acceptance by the V&V team.

5) No changes, subsequent to the request for entry or acceptance, is made without authorization by the chief programmer.

6) The librarian shall assign an official identification number to the module or subsystem and files the corresponding source listing notifying the chief verifier that the software item is ready for validation.

7) When the V&V is satisfied that the software item has met specifications, the chief verifier releases it back to the library as acceptable for use.

8) After ensuring the proper compliance with the Software Configuration Management Plan's (SCMP) procedures, by both the designer and verifier, the Librarian will:

   a. catalog the software item assigning a revision number and

   b. file the relocatable object code, and related documentation.

9) All further revisions shall increment the revision number and shall be documented in sufficient detail to show the transition from one revision to the next.

10) All revisions shall be approved by the chief verifier.

11) Details of library functions, access, change controls and other operational procedures shall be provided in the SCMP.

4.9.5.1 Software Librarian

1) The Software Librarian is responsible for the maintenance of the software library and will perform the following typical duties:

   - Ensure that both the design and V&V teams comply with the appropriate procedures specified in the SCMP.

- Retain records of the software modules and the associated documentation in terms of usage and revision levels.
- Provide copies of source code listings, and other associated documentation to library users upon request.
- Maintain a collection of verified software which is used in higher level systems and other projects.

5.0 SOFTWARE CONFIGURATION MANAGEMENT PLAN

5.1 INTRODUCTION

5.1.1 Purpose

The purpose of this section is to describe the methodology used in managing the control of software for the Nuplex 80+ system.

This control will be accomplished by:

- Maintaining historical records of software as it is developed and maintaining the relationship between that software and the corresponding functional documents.
- Controlling Changes to Software
- Recording and reporting change processing and implementation status.

The software configuration management forms described in this section are typical. NUPLEX 80+ systems may employ automatic configuration management software to support the requirements of this section. Such packages may produce documentation that is different in format but meets the intent of the forms in this section.

5.1.2 Scope

All software items and associated documentation shall be controlled in such a manner as to maintain the items in a known and consistent state at all times. Original software and modifications to existing software shall follow the configuration requirements for all life cycle phases. Existing software which is not modified shall only fall under configuration control procedures upon its introduction into the software systems.

5.2 MANAGEMENT

5.2.1 Organization

All software configuration management functions for a system are performed by the supervisory group responsible for a system. V&V activities related to configuration management are performed by member(s) of a V&V team, outside of the supervisory group for the system.

5.2.2 Software Configuration Management Goals

For each system, the SCMP shall establish the objectives that it intends to satisfy in order to support the realistic needs of the software life cycle. The following objectives should be considered in developing system specific SCMP's:

- To record and document the work in progress on each software item to facilitate understanding of current project status.
- To identify all software code (modules and units) and data associated with a system, including revision level, completion status, test status and history.
- To record the work in progress on any software item to facilitate understanding of current project status.
- To maintain the association among software documents, code and data.
- To identify sets of software items that compose the system (baselines), test status and history, and readiness for release.
- To maintain the status of each released baseline, customer lists, and associated software problem reports.
- To support the evaluations of impact of a software problem on other baselines and customers.
- To maintain the association among software problem reports, change reports and the affected documents sections, lines of code and data items.

- To implement the appropriate controls and approvals for any changes to the software configuration.
- For safety critical software, identify all personnel with a given software item, problem report, or software change.
- To document the criteria for the generation of releasable baselines.
- To assure that designated prior revisions of individual software items and baselines can be reconstituted in the future.
- To backup software (completed and in progress) to protect against disaster.
- To plan for controlling access to computer software and protecting against software viruses.

5.2.3 Software Configuration Management Responsibilities

The supervisor of the group responsible for a system is responsible for the implementation of adequate measures to manage and control the software configuration for the system, consistent with this software configuration management plan.

Within each supervisory group responsible for a system, a librarian maintains all controlled software items as well as all control records. The librarian also maintains multiple backups of these items at alternate locations to assure sufficient disaster recovery capability. The librarian may be a member of the design or requirements team for the system, but reports to the supervisor responsible for the system.

A single librarian may be used for more than one system under the cognizance of more than one supervisor.

A system administrator, who may also be the librarian, is responsible for performing ongoing backup of work in progress for the system.

5.2.3.1 Configuration Identification

Throughout the software life cycle, all versions of all software items shall be tracked along with their associations. For example, the association of a particular version of a code item and related design and requirements documents, test reports, etc. will be maintained in lifetime software records per reference 1.6.11. The association of software items with each particular software baseline shall also be maintained in lifetime software records. The scheme by which this is done will be developed for each specific system by the CEO at the start of software development. The scheme is subject to the approval of Nuplex 80+ project management.

The scheme shall associate items listed in section 5.2.3.2. Some of these items have specific numbering methods, also described in section 5.2.3.2.

5.2.3.2 Configuration Item Numbering Method

Each software item shall have a numbering method that shall uniquely identify the item and its versions. The configuration identification scheme in section 5.2.3.1 will be consistent with the numbering method for all software items. Items requiring such a method include those described below. Unless a specific method is identified in this section for the item, the CEO shall determine the numbering method for each specific system at the start of software development.

- Software documents are identified by a character string appearing on the cover. The format for this identifier shall be consistent and compatible with the Nuclear Engineering Documentation System (Reference 1.6.10).

- Source and object code computer files for modules are identified by a unique module name defined in the associated software requirements document for the system, and a revision identifier. This label must appear in a comment section at the beginning of the module source code, as follows:

  XXXXXXXX-FF-MM-RR

  where:

XXXXXXXX	is the module name
FF	is a sequential number for successive versions that implement the requirements in a new or revised software requirements specification.
MM	is a sequential number for successive versions that implement changes to a software design description (reset to 00 when FF changes).
RR	is a sequential number for successive versions requiring no change to the requirements specification or design description (usually coding error corrections). (Reset to 00 when FF or MM changes)

- Program units and system baselines
- Data for execution
- Software quality assurance records
- Other software materials (test data, results)

All software and documentation are released to and maintained by the CEO in a controlled library. The CEO administers the change control process.

Each above item, and each revision, that is entered into the managed configuration is uniquely identified. Software documents are identified by a string of characters appearing on the cover of each document. This character string shall be consistent and compatible with the Nuclear Engineering Documentation System (Reference 1.6.10).

5.2.3.2.1 Software Change Request (SCR)

The SCR is the mechanism by which change requests are approved by the CEO supervisor and presented to the librarian. This action allows a developer to check out software/documentation from SCM controlled libraries. An SCR may be generated from an action documented in a Test Exception Report (TER). Refer to Section 8.2 for a description of Problem Reporting. The mechanism for requesting authorization is to present the SCR to the CEO supervisor and request approval for work to begin. The SCR form shown in Exhibit 5-1 is to be used.

5.2.3.2.2 Software Change Authorization (SCA)

The SCA is used to request the placement of a new version or changes of software/documentation into the controlled libraries. The SCA form shown in Exhibit 5-2 is to be used.

5.2.3.2.3 Software Module Release History

The documentation of software module release history is performed by completing Exhibit 5-3. This will enable the configuration management program to maintain an accurate status of each software module at all times and enable the reconstruction of a given release at any time in the future.

5.2.3.2.4 Software Unit Release History

The documentation of software unit release history is performed by completing Exhibit 5-4. A software unit may be made up of several software modules which may have various module release dates. However, each software unit release will have the latest revision of all modules as of the date of the unit release. This is to insure that the release can be reconstructed in the future by maintaining an accurate configuration status at all times.

5.2.3.3 Status Accounting

A software change authorization data base is used for generating reports that track changes to all the controlled baselines. The CEO generates reports that track the status of documentation and the software.

5.2.3.4 Baselines

Baselines are established for the control of design, product, and engineering changes and are time phased to the development effort. Baselines are established by each CEO to capture a software stage in the development effort. The Baseline Release History form Exhibit 5-5 is required to be completed to document the date of the baseline and to identify the users of that specific release.

5.2.3.5 Backups

Each CEO shall perform backups of all electronic media. Backups shall be performed daily while software is under development. At a minimum, daily backups shall be maintained for one month. Monthly backups shall be maintained for one year with the exception of safety related software for which monthly backups will be maintained indefinitely. After development is completed, released versions of the software shall be archived permanently. The media and contents of this archive shall be periodically revalidated at a frequency to be determined by the CEO (i.e., once per year).

5.2.3.6 Configuration Control Responsibility

The CEO supervisor has responsibility for policies, plans, procedures and interfaces within that CEO. The software change request that is generated is reviewed by the CEO supervisor and one of the following actions taken: approved, disapproved, or tabled.

5.2.4 Interface Control

Interface Control is handled in the same manner as other types of software or documentation. Any differences between the SQAP and the SCMP must be resolved prior to the establishment of any baselines.

5.2.5.2 Configuration Baselines

Baselines are established by the acceptance of the Nuplex 80+ system specifications. This occurs at the completion of the Nuplex 80+ system requirement review (SRR). New baselines may be established at the discretion of the CEO supervisor after one or more software change requests have been implemented.

5.3 SOFTWARE CONFIGURATION MANAGEMENT ACTIVITIES

5.3.1 Configuration Identification

All software and documentation shall be uniquely identified by a system name, number, and corresponding revision date, appropriate category and appropriate software classification. Reviews, audits, problem reports, and test reports shall specifically identify the system name, number, revision number, and revision date to minimize confusion. This identification structure shall also have the ability to track resolution of test results and subsequent retesting of software modules and integrated systems. Lists of references which may be included within the documentation shall likewise provide identification data for all references to other products used within the project.

This configuration identification data shall be defined as part of the baseline at the completion of each major phase of the software project. In order for the baseline to change, it must incorporate all approved changes and represent the most recent approved software configuration.

5.3.2 Configuration Change Control

Following the original baseline after the Implementation Phase software and documentation shall be placed under configuration control by the CEO group. Configuration change control by the CEO group shall continue up to the point when the software is no longer expected to be changed by the CEO. Software configuration control shall include limiting access to master copies of documentation and software in either hard copy or magnetic form. Access to software and documentation shall be controlled by the CEO group librarian.

The CEO Librarian shall implement a procedure to detect and eliminate any software viruses.

After the software is "System Level Certified" by the CEO, it shall be delivered for integration testing.

The SCMP shall address how software changes will be managed. There shall be three environments which logically separate all software placed under configuration control:

- The development environment is where all software is

written, changed, and unit tested by the CEO group and controlled by the CEO librarian.

- The control library is where all master copies of software modules reside for integration test purposes, and are controlled by the librarian.

- The target library is where all final configured, tested and integrated software will reside.

Changes to software shall be formally documented. This documentation shall contain a description of the change, the rationale for the change, and the identification of affected software element(s). Software verification and/or validation shall be performed on the change, in accordance with the SVVP, to ensure the necessary traceability, consistency, and correctness of the change.

Details of this control procedure shall be given in the system specific SCMP. (The methods of configuration control may vary from those described above if an automated configuration management package is used. In this case the SCMP will describe the procedure from the programmer and librarian perspective.)

5.3.3 Configuration Status Control Log

A software configuration status control log shall be maintained by the CEO group which shall show a record of the revision history and current status of each controlled software element. This log shall include:

- Identification of approved baseline configuration

- Status of proposed changes

- Status of approved changes

An automated configuration management system may be incorporated on this project. If this is the case, the features of the configuration log described above shall be included in such a tool.

6.0 SOFTWARE OPERATION AND MAINTENANCE PLAN

6.1 PURPOSE

To specify the requirements for the maintenance and use of computer software utilized for nuclear safety related functions.

6.2 OPERATIONS AND MAINTENANCE PHASE

6.2.1 The CEO supervisor is responsible for coordinating activities necessary to maintain the software, to remove latent errors, respond to a new or revised requirements or to adapt the software to changes in the operating environments. Software modifications shall include the activities of the development cycle phases as appropriate, including verification and validation. These activities and their documentation shall be limited to those portions of the program that are actually modified except that the validation phase shall include testing as necessary to verify that the modifications have not caused unintended adverse effects.

6.2.2 The CEO supervisor shall be responsible for ensuring that errors reported by users or the software developer are evaluated, documented and reported to other users, and to the software developer for correction if required.

The Test Exception Report (Exhibit 8.1) is prepared to document a problem with the system. If this problem is determined by the CEO to be a defect in the software, the CEO shall notify all users that are affected by the defect.

1) The distribution list of the managers of groups who use the software shall be maintained up-to-date. The list shall include the appropriate Project or task Manager responsible for notifying external client users of CENSYS software products or services.

2) The CEO supervisor shall distribute the Computer Program Error Notification form, similar to Exhibit 6-1 to each of the individuals on the distribution list.

3) Each CENSYS Manager receiving the Computer Program Error Notification shall ensure that all users of the program in his group are advised of the error. Each CENSYS user shall assess the impact of the code error on work for which the user has utilized the code, and complete the Response Receipt, Exhibit 6-2. If the user determines that the error impact is "Yes", the user is responsible for taking and documenting appropriate action as necessary including 10CFR50 and/or 10CFR21 considerations. Documentation shall include or reference the Computer Program Error Notification.

4) The completed Computer Program Error Notification form shall be returned by the user group manager to the CEO Manager within sixty (60) days. The CEO supervisor shall ensure that all forms have been returned and submit the package to be retained as quality records per Reference 1.6.11.

Project/Task Managers responsible for notifying external users shall return the Response Receipt with evidence that notice was sent to the user.

6.3 RETIREMENT PHASE

6.3.1 When a software product is no longer needed or has been superseded by a later revision, the CEO supervisor shall notify users in writing of the termination of the Computer Code Certificate and shall take necessary steps to prevent access to the program.

7.0 DOCUMENTATION

7.1 GENERAL REQUIREMENTS

Software documentation shall be provided for all computer software to be used or delivered for the Nuplex 80+ system.

7.2 FUNCTIONAL REQUIREMENTS DOCUMENTS

There are no specific requirements for format and content of documents that describe the system's functional requirements. However, the interface between functional requirements document(s) and the design documents will be subject to quality assurance provisions. Functional requirements documents include:

- Operational Requirements
- Performance Requirements
- Theoretical Basis
- Design Constraints
- External Interfaces

7.3 SOFTWARE DESIGN REQUIREMENTS (SDR)

The SDR is developed according to IEEE 830-1984 and is used as the source document for design of the software, including:

1) Description of major software components which reflect the software requirements

2) Technical description of the software (i.e. control flow, data flow, control logic, data structures)

3) Description of all interfaces and allowable ranges of inputs and outputs

4) Any other design items which must be translated into code

5) A description of the intended platform and programming language(s) expected to be utilized

6) Data necessary for final implementation such as setpoints

7) Abnormal conditions to be accommodated by the software shall be described, including resulting functional operations.

8) Plant input signal transient conditions to be accommodated by this software shall be described.

Each requirement in the SDR shall be defined such that its achievement is capable of being verified by the SVVP. The SDR shall specify any V&V requirements that must be included in the system specific SVVP.

7.4 SOFTWARE DESIGN DESCRIPTIONS (SDD)

The software design description is prepared according to IEEE 1016-1987.

The purpose of the SDD is to depict how the software will be structured to satisfy the requirements of the SDR. The design shall be described such that it can be translated into software code.

The Software Design Description is a detailed description of the software to be coded. It describes decomposition of the software into entities. Each entity is described by its type, purpose or function, subordinate entities, dependencies, interfaces, resources, processing and data.

Each design feature shall be described and defined such that its achievement is capable of being verified and validated per the SVVP. The adequacy of the SDD shall be verified against how the requirements of the software (documented in the SDR) are to be implemented in code, and how the design is traceable to the requirements in the SDR.

7.5 SOURCE CODE DOCUMENTATION

Source code documentation shall include source code listings which reflect the translated design representation into a programming language. Also, any associated documentation generated during the coding, module testing, and unit testing process such as code reviews, test procedures, test cases or test results which are not required by Exhibit 3-5, shall be included with the source code listings.

Source code shall be traceable to the software design documented in the SDD and the requirements in the SDR. It shall include sufficient comments to provide the user of the source code with an understanding of the functioning and programming of each module. All source code, whether developed or modified from existing software, shall be documented in accordance with the coding standards listed in Section 3.4.2.1.

7.6 SOFTWARE VERIFICATION & VALIDATION DOCUMENTATION

Software V&V documentation shall include system specific Software V&V Plans (SVVP) and Software V&V Reports (SVVR), prepared according to IEEE 1012-1986.

7.6.1 Software Verification and Validation Plan

The SVVP shall identify the software items to be evaluated and shall describe the V&V evaluation and reporting activities. Verification review requirements and guidelines are described in Section 3.5. Validation tests to be performed shall be described in a separate Test Plan which is subordinate to the SVVP, and is included as part of the software V&V documentation.

For custom software to be developed, the SVVP shall be developed during the Software Requirements Phase (Section 3.2.2.3) based on the specific V&V requirements set forth in the SDR.

For existing software to be modified, the system specific SVVP shall include methods for verifying and validating modifications to this existing software.

The SVVP shall provide adequate planning for the following:

- Software V&V process for the various software categories described in Section 3.1.1
- Software V&V process for existing software to be modified and to be used "as-is".
- Software V&V process for prototype software

The SVVP shall also facilitate the tracking and recording of the hardware configuration pertinent to the software verification and validation process during all phases of the software life cycle.

7.6.2 Software Verification and Validation Report

A V&V file shall be maintained by the CEO throughout the software life cycle to document all V&V activities. It shall be an ongoing compilation of all validation test results, problem reports and corrective actions (Section 8.0), verification review results (Section 3.5), and the results of all NQA audits (Section 3.5). This file shall form the basis for the development of a system specific SVVR upon site installation and checkout. The V&V report shall have a cover sheet similar to Exhibit 7-1.

Each SVVR shall be developed by the CEO. It shall also include sections for the following:

- Summary of tests performed and results
- Test Procedures
- Test Results (including all Test Exception Reports described in Section 8)

7.7 USER DOCUMENTATION

User documentation is prepared according to IEEE 1063-1987. The purpose of User Documentation is to provide sufficient information about the software to permit users to employ the code as it was intended. It shall be written by the design team and verified by the V&V team within the CEO group where the software was developed. User documentation shall be developed during the Test Phase and delivered to the user during the Installation and Checkout Phase.

User documentation shall consist of vendor documents and documents prepared as part of the project. Project prepared user documents shall be as follows:

- User's Manual
- Installation and Operations Manual
- Maintenance Manual

User Documentation shall include all error messages and identify the necessary corrective-action procedures. Also, it shall provide the means for the user to report problems to the CEO group or software maintenance group.

7.8 SOFTWARE CONFIGURATION MANAGEMENT DOCUMENTATION

Each group within the CEO shall be responsible for developing its own SCMP, prepared according to IEEE 828-1990 to satisfy the unique requirements of their software development process. It shall address the following areas as a minimum:

- Description of software covered
- Change control procedures
- Documentation required
- Baseline identification and configuration
- Virus Prevention
- Access Control

The SCMP for each CEO group shall cover the following phases of software life cycle:

- Software Requirements
- Software Design
- Software Implementation

A SCMP shall also be developed by the CEO to cover the following phases:

- Testing
- Site Installation and Checkout

7.9 COMPUTER CODE CERTIFICATE

The completion of the Software Verification and Validation report is the basis for the issuance of a Computer Code Certificate, Exhibit 7-2.

Computer Code Certificates are issued for safety-related software only.

The issuance of a Computer Code Certificate allows the release of a configuration item for use in a safety-related application.

Software intended for limited use, such as in a single design analysis, may be used provided that the results as well as methods and/or formulas are documented in the design analysis in sufficient detail to allow independent verification. A Computer Code Certificate shall not be issued for such software on this basis alone.

8.0 PROBLEM REPORTING AND CORRECTIVE ACTION

8.1 CLASSIFICATION SYSTEM

A five-level priority scale shall be used in the classification of software problems (Exhibit 5-1). The responsibility of assigning software problems found during review or testing to one of these classes shall lie with the independent reviewer or tester together with cognizant engineer.

8.2 PROBLEM REPORTING

Discrepancies, deficiencies, or comments identified as a result of testing, review, or other means shall be documented in a formal manner. This includes any general discrepancies found outside of the normal V&V test process. The following table illustrates the type of report required by each method:

METHOD	REPORT
Verification Reviews (per Section 3.5)	Comment Records (CR) (Exhibit 3-6)
Validation Tests	Test Exception Report (TER) (Exhibit 8-1)
General Findings	Test Exception Report (TER) (Exhibit 8.1)

The appropriate configuration identification data (see Section 5.3.1) for each deficient software item or document shall be included on the appropriate form (or report). The form (or report) shall also include a description of the observed deficiency, the name of the individual reporting the deficiency, and the date of the report finding.

In the case of a TER, each form shall include space for a description of the resolution and any retest or review required after the resolution. If retest is performed, a copy of the test procedure or test case used shall be attached to the completed TER. Copies of all completed TER forms shall be distributed by the independent reviewer to CEO management and placed under configuration management per the system specific SCMP. All TER forms shall also be placed in a lifetime records per reference 1.6.11. The steps taken to cause the discrepancy to occur should also be included on the TER form in order to reproduce the problem. These steps should be noted as best as possible if the problem is not repeatable.

The extent of the retest shall be determined by the CEO group based on the relative impact of the software change on the overall system operation. For Protection and Important to Safety software, all changes require complete system or function retest, unless otherwise justified in writing including steps to ensure that new errors were not introduced.

8.3 CORRECTIVE ACTION

The CEO shall establish as a clear objective the goal of resolving all validation test problems (via TER's) and verification review comments (via CR's) expeditiously to minimize the potential for unidentified effects during later life cycle phases.

The corrective action procedures used shall be based on the level of problem reported.

In addition, the CEO shall adhere to the following corrective action methodology that:

- Problems are identified, evaluated, documented and, if required, corrected by the appropriate reporting mechanism (Section 8.2).
- Problems are classified per the five-level priority scale.
- Corrections or changes shall be controlled per the appropriate sections of the SCMP.
- Preventive actions and corrective actions are documented on the appropriate form and distributed to the cognizant CEO group.

Corrective actions shall be documented on TER's and CR's by the cognizant engineer and shall be completed by the due date specified on the form. If a resolution is not received by the specified due date, the problem shall be escalated to the next level of CEO management. This escalation process is continued through each higher level of CEO management until a response is received. Once the independent reviewer is satisfied with the corrective action taken, the report form shall be signed off and entered into the report form central log. (It should be noted that this process may not be necessary for all software problems. The CEO shall use the severity classification system described in Section 8.1 to determine which classes of software need to be handled through the escalation process.)

9.0 USNRC RESTRICTIONS ON REVISIONS TO THE SOFTWARE PROGRAM MANUAL

The United States Nuclear Regulatory Commission (USNRC) staff has determined that the certain sections of the Software Program Manual cannot be changed without prior USNRC approval, because these sections contain basic information relied on by the staff for its safety determination.

Within the following major section:		Modifications to following sections not allowed without prior USNRC approval:	
1.0	Introduction	1.1 1.2 1.3 1.4	Purpose Scope Overview General Requirements
3.0	Software Quality Assurance Plan	3.1 3.2 3.3 3.5 3.6 3.8	Purpose Management Documentation Review Requirements Problem Reporting and Corrective Action Code Control
4.0	Software Verification and Validation Plan	4.1 4.2 4.8	Purpose Overview of V&V System Validation Testing
5.0	Software Configuration Management Plan	5.1 5.2	Introduction Management
6.0	Software Operations and Maintenance Plan	6.1	Purpose
7.0	Documentation	7.1 7.2 7.3 7.4 7.6 7.9	General Requirements Functional Requirements Documents Software Design Requirements Software Design Description Software Verification & Validation Documentation Computer Code Certificate
8.0	Problem Reporting and Corrective Action	8.1 8.2 8.3	Classification System Problem Reporting Corrective Action
9.0	USNRC Restrictions on Revisions to the Software Program Manual	9.0	USNRC Restrictions on Revisions to the Software Program Manual

Any changes to the commitments identified in the sections above, except for editorial changes and layout, could involve an unreviewed safety question, and therefore will require NRC review and acceptance prior to implementation. Any requested changes to these commitments shall be specifically described in the Combined Operating License (COL) application or submitted for license amendment after COL issuance.

EXHIBIT 1-1
SOFTWARE PROGRAM DOCUMENTATION HIERARCHY

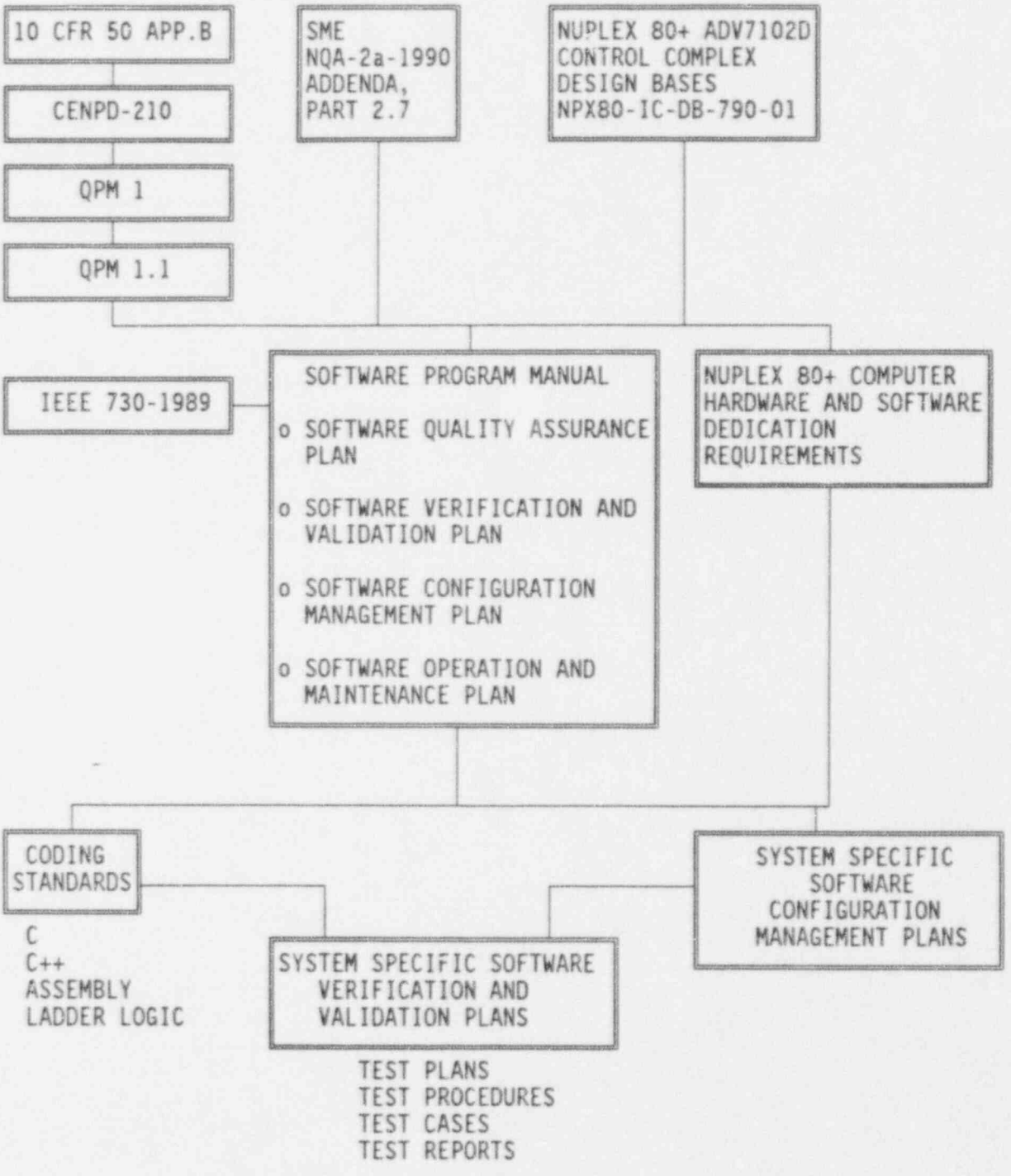


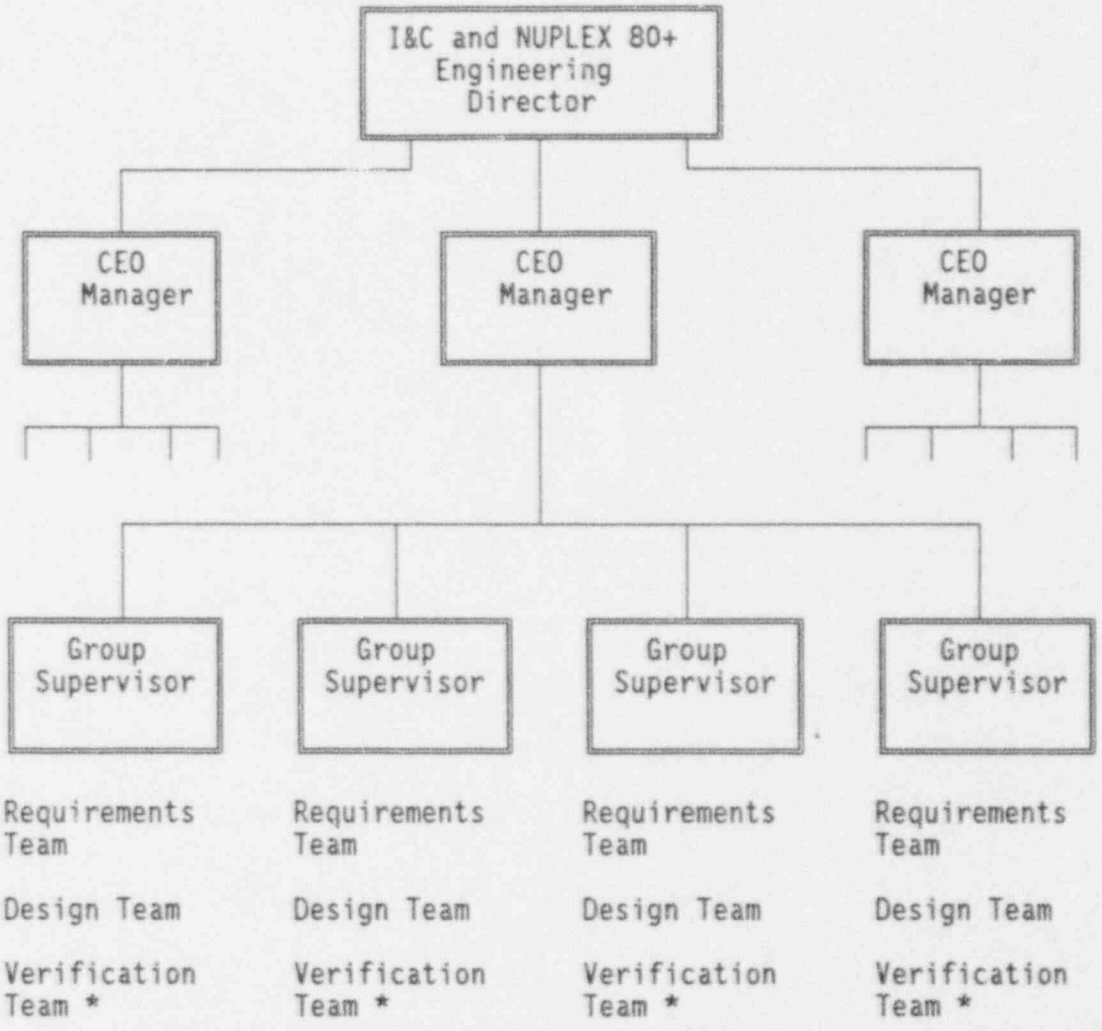
EXHIBIT 1-2

EXAMPLE OF A FILLED OUT SOFTWARE CLASS/CATEGORY MATRIX
FOR ONE TYPE OF NUPLEX 80+ SYSTEM

	SOFTWARE CLASSES			
SOFTWARE CATEGORIES	PROTECTION	IMPORTANT TO SAFETY	IMPORTANT TO AVAILABILITY	GENERAL PURPOSE
Original	None	Special Applications for Safety System Monitoring	Data Base New Applications Display Formats	None
Existing to be Modified	None	None	Applications from Previous Projects Display Formats	None
Existing not to be Modified	None	None	Operating Systems Compilers	None

EXHIBIT 2-1

NUPLEX 80+ SYSTEM AND SOFTWARE DEVELOPMENT ORGANIZATION



* For System(s) whose design and requirements are developed under other group supervisors.

Exhibit 3-1
ASSIGNMENT OF SYSTEMS AND SUB-SYSTEMS TO CLASSES

SYSTEM	SUB-SYSTEM SCOPE	CLASS
Data Processing System (DPS)	Safety Parameter Display System (SPDS) Algorithms	Important to Safety
	Programs Not Required for Operation	General
	All Other DPS Software	Important to Availability
Post Accident Monitoring Instrumentation (PAMI)	All Software	Important to Safety
Discrete Indication and Alarm System (DIAS)	PAMI System Related Displays	Important to Safety
	All Other DIAS Software	Important to Availability
Integrated Process Status Overview	All Software	Important to Safety
Power Control System (PCS)	Limiting Functions Software	Important to Safety
	All other Software	Important to Availability
Engineered Safety Features-Component Control System (ESF-CCS)	Safety Critical Kernel	Protection
	Man-Machine Interface (MMI) Test Processor Software	Important to Safety
	All other Software	Important to Safety
Process Component Control System (PCCS)	Software necessary to perform alternate protection system control actions	Important to Safety
	All other Software	Important to Availability
Nuclear Integrity Monitoring System	All Software	Important to Availability
Plant Protection System (PPS)	Safety Critical Kernel	Protection
	MMI test Processor Software	Important to Safety
	Inter-System Communication Software	Important to Safety
	All Other Software	Important to Safety

This table should be used as guidance for other Nuplex 80+ systems and functions.

EXHIBIT 3-2
Software Class and Category Worksheet

System:

Subsystem Class (circle):

Protection Important to Safety Important to Availability General Purpose

Subsystem Scope Description:_____

_____

_____

_____

_____

Software Item	Check Applicable Box		
	Original	Existing, to be Modified	Existing, not to be Modified

Prepared By: _____ Date: _____

Approved By
Group Supervisor: _____ Date: _____

EXHIBIT 3-3
TYPICAL NUPLEX 80+ SOFTWARE DEVELOPMENT PROCESS

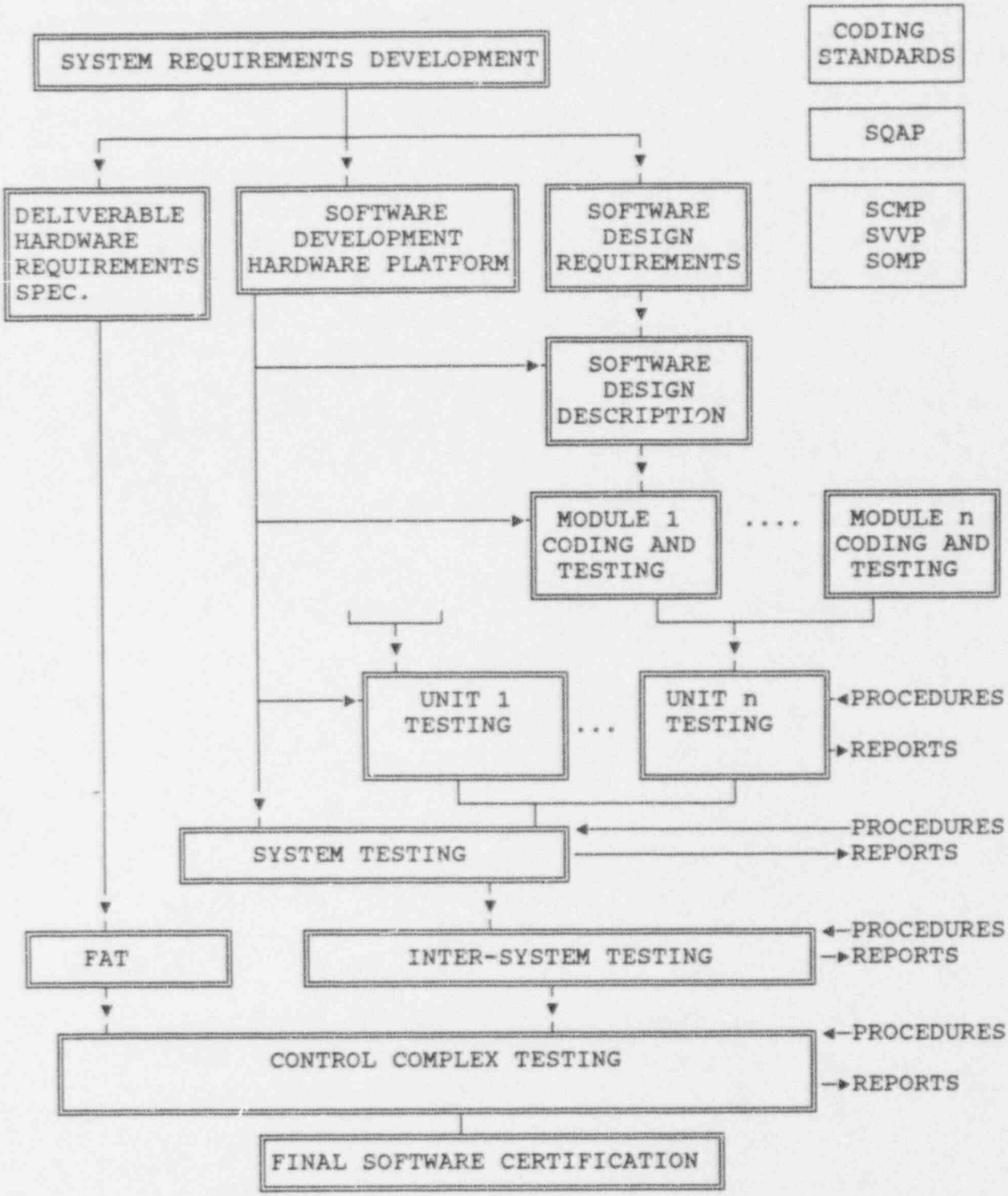


EXHIBIT 3-4: TASKS REQUIRED FOR SOFTWARE CATEGORIES

TASK	ORIGINAL SOFTWARE	ETBM SOFTWARE	ENM SOFTWARE
SQA PLANNING PHASE			
SOFTWARE QUALITY ASSURANCE PLAN	X	X	X
CODING STANDARDS	X	X	
SYSTEM SPECIFIC SOFTWARE VERIFICATION AND VALIDATION PLAN	X	X	X
GROUP SPECIFIC SOFTWARE CONFIGURATION MANAGEMENT PLAN	X	X	X
FUNCTIONAL REQUIREMENTS PHASE			
SYSTEM FUNCTIONAL REQUIREMENTS	X	X	X
PROTOTYPE CODING	As Required		
SOFTWARE REQUIREMENTS PHASE			
SOFTWARE DESIGN REQUIREMENTS	X	X	X
SOFTWARE DESIGN PHASE			
SOFTWARE DESIGN DESCRIPTION	X	X	
SOFTWARE IMPLEMENTATION PHASE			
MODULE CODING	X	X	
TEST PLAN (started in the requirements phase)	X	X	X
MODULE TEST EXECUTION	X	X	
UNIT TEST PROCEDURE (Protection and Important to Safety)	X	X	X
UNIT TEST EXECUTION	X	X	X
UNIT TEST CERTIFICATE	X	X	X

ETBM - Existing To Be Modified
ENM - Existing Not To Be Modified

EXHIBIT 3-4: TASKS REQUIRED FOR SOFTWARE CATEGORIES

TASK	ORIGINAL SOFTWARE	ETBM SOFTWARE	ENM SOFTWARE
TESTING PHASE			
SYSTEM TEST PROCEDURE (started during requirements phase)	X	X	X
SYSTEM TEST EXECUTION	X	X	X
SYSTEM TEST REPORT	X	X	X
INTER-SYSTEM TEST PROCEDURE (started during requirements phase)	X	X	X
INTER-SYSTEM TEST EXECUTION	X	X	X
INTER-SYSTEM TEST REPORT	X	X	X
INTEGRATION TEST PROCEDURE	X	X	X
INTEGRATION TEST EXECUTION	X	X	X
CONTROL COMPLEX INTEGRATION TEST REPORT	X	X	X
USER DOCUMENTATION	X	X	X
SITE ACCEPTANCE TEST PROCEDURE (started during requirements phase)	X	X	X
SITE INSTALLATION AND CHECKOUT			
SITE ACCEPTANCE TEST EXECUTION	X	X	X
SITE ACCEPTANCE TEST REPORT	X	X	X
OPERATION AND MAINTENANCE			
MAINTAIN SOFTWARE	X	X	X
RETIREMENT	X	X	X

ETBM - Existing To Be Modified
ENM - Existing Not To Be Modified

EXHIBIT 3-5

UNIT TESTING CERTIFICATE

This certificate should be filled out by the software designer for each program module upon completion of unit testing. Copies of written notes and documentation that have been maintained during unit testing should be attached to this form.

Unit Name:_____ Version:_____

1. Description of Incremental Test performed.

2. Description of Code/Data Inspection performed.

3. Description of Functional Test performed.

4. Description of unresolved errors. Other comments.

Prepared by:_____ Date:_____

Reviewed by:_____ Date:_____

EXHIBIT 3-6
NUPLEX 80+ COMMENT RECORD

DOCUMENT NUMBER:		
ITEM NO.	COMMENT	COMMENT RESOLUTION
REVIEWER: _____ Signature _____ Date		DOCUMENT ORIGINATOR: _____ Signature _____ Date
		RESOLUTION REVIEW: _____ Signature _____ Date

EXHIBIT 4-1: SOFTWARE TASKS AND RESPONSIBILITIES

TASK	PROTECTION	IMPORTANT TO SAFETY	IMPORTANT TO AVAILABILITY	GENERAL
SQA PLANNING PHASE				
SYSTEM SPECIFIC SVVP'S	VT/DT	VT/DT	VT/DT	VT/DT
SYSTEM SPECIFIC SCMP'S	DT/VT	DT/VT	DT/VT	DT/VT
FUNCTIONAL REQUIREMENTS PHASE				
SYSTEM FUNCTIONAL REQUIREMENTS	RT	RT	RT	RT
PROTOTYPE CODING	DT	DT	DT	DT
SOFTWARE REQUIREMENTS PHASE				
SDR	DT/(VT & RT)	DT/(VT & RT)	DT/(VT & RT)	DT/VT OR RT
SOFTWARE DESIGN PHASE				
SDD	DT/VT	DT/VT	DT/VT	DT
SOFTWARE IMPLEMENTATION PHASE				
MODULE CODING	DT/VT	DT/VT	DT/DT	DT
TEST PLAN (MAY BE PART OF SVVP)	VT/DT	VT/DT	VT/DT	DT
MODULE TEST EXECUTION	DT	DT	DT	DT
UNIT TEST PROCEDURE	VT/DT	DT/VT	N/A	N/A
UNIT TEST EXECUTION	VT	DT	DT	DT
UNIT TEST CERTIFICATE	VT/DT	DT/VT	DT/VT	DT/VT
KEY: ORIGINATOR/REVIEWER (E.G. DT/VT)	DT = DESIGN TEAM VT = V&V TEAM RT = REQUIREMENTS TEAM			

EXHIBIT 4-1: SOFTWARE TASKS AND RESPONSIBILITIES

TASK	PROTECTION	IMPORTANT TO SAFETY	IMPORTANT TO AVAILABILITY	GENERAL
TESTING PHASE				
SYSTEM TEST PROCEDURE	VT/DT	VT/DT	DT/DT	DT
SYSTEM TEST EXECUTION	VT	VT/DT *	DT/VT *	DT
SYSTEM TEST REPORT	VT/DT	VT/DT	DT/VT	DT
INTEGRATION TEST PROCEDURE	VT/(DT & RT)	VT/(DT & RT)	VT/(DT & RT)	VT/(DT & RT)
INTEGRATION TEST EXECUTION	VT	VT	VT	VT
INTEGRATION TEST REPORT	VT/(DT & RT)	VT/(DT & RT)	VT/(DT & RT)	VT/(DT & RT)
USER DOCUMENTATION	DT/VT	DT/VT	DT/VT	DT/VT
SVVR	VT/DT	VT/DT	VT/DT	VT/DT
SAT PROCEDURE	DT/VT	DT/VT	DT/VT	DT/VT
SAT REPORT	VT	VT	VT	VT
KEY: ORIGINATOR/REVIEWER (E.G. DT/VT)	DT = DESIGN TEAM	VT = V&V TEAM	RT = REQUIREMENTS TEAM	

* VT may witness system test execution at their discretion

Any single software item may be, at the discretion of the CEO, handled according to a class that is more important to safety. For example, a particular important to safety software item may be, at the discretion of the CEO, unit tested by a V&V team without redesignating the software item as protect class.

EXHIBIT 4-2
CHECKLIST NO. 1

SOFTWARE VERIFICATION AND VALIDATION
REQUIREMENT PHASE CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

REQUIREMENT PHASE	Yes	N/A
Are the requirements complete?	___	___
Are the requirements correct?	___	___
Are the requirements internally consistent?	___	___
Are the requirements clear and unambiguous?	___	___
Are the requirements feasible?	___	___
Are the requirements testable?	___	___

Reviewer's comments (Optional): _____
_____

Reviewed by: Name _____ _____ Signature _____ Date _____

EXHIBIT 4-3
CHECKLIST NO. 2

SOFTWARE VERIFICATION AND VALIDATION
DESIGN PHASE CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

DESIGN PHASE	Yes	N/A
Is the design traceable to the requirements?	___	___
Is the design complete?	___	___
Is the design correct?	___	___
Is the design internally consistent?	___	___
Is the design clear and unambiguous?	___	___
Is the design feasible?	___	___
Is the design testable?	___	___

Reviewer's comments (Optional): _____
_____

Reviewed by: Name _____ Signature _____ Date_____

EXHIBIT 4-4
CHECKLIST NO. 3

SOFTWARE VERIFICATION AND VALIDATION
IMPLEMENTATION PHASE CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

IMPLEMENTATION PHASE	Yes	N/A
Are the comment statements provided sufficient to give an adequate description of each routine and data structure?	___	___
Is the source code understandable	___	___
Is the source code consistent with the design?	___	___
Are all the variables properly specified and used?	___	___
Is there satisfactory error checking?	___	___
*Do all subroutine calls transfer data variables correctly?	___	___
Is the data read from each file consistent with the data written to that file?	___	___

Reviewer's comments (Optional):_____
_____

Reviewed by: Name _____ Signature _____ Date _____

EXHIBIT 4-5
CHECKLIST NO. 4

SOFTWARE VERIFICATION AND VALIDATION
TEST PHASE CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

TEST PHASE	Yes	N/A
Test Plan		
Is the Test Plan description complete?	___	___
Are the test problem definitions adequate and complete?	___	___
Is each testable requirement adequately covered?	___	___
Is the plan for evaluating and reporting test results adequate?	___	___
*For program maintenance changes, have adequate regression tests been specified to verify that the modifications have not caused adverse effects on unmodified code?	___	___

Reviewer's comments (Optional): _____
_____

Reviewed by: Name _____ _____ Signature _____ Date _____

EXHIBIT 4-6
CHECKLIST NO. 5

SOFTWARE VERIFICATION AND VALIDATION
TEST RESULTS REPORTING CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

Test Results Reporting	Yes	N/A
Do the test results comply with the format specified in the Test Plan?	___	___
Do the test results provide and accurate statement of the testing performed?	___	___
Have test results been evaluated as acceptable?	___	___

Reviewer's comments (Optional): _____
_____

Reviewed by: Name _____ Signature _____ Date _____

EXHIBIT 4-7
CHECKLIST NO. 6

SOFTWARE VERIFICATION AND VALIDATION
INSTALLATION AND CHECKOUT PHASE CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

INSTALLATION AND CHECKOUT PHASE	Yes	N/A
User Documentation		
Is the information in the User Documentation consistent with that in the Requirements?	___	___
Is the information in the User Documentation consistent with that in the Design?	___	___
Is the information in the User Documentation an accurate reflection of the coded program?	___	___
Is the information in the User Documentation internally consistent?	___	___
Is the information in the User Documentation clear and unambiguous?	___	___

Reviewer's comments (Optional): _____
_____

Reviewed by: Name _____ Signature _____ Date _____

EXHIBIT 4-8
CHECKLIST NO. 7

SOFTWARE VERIFICATION AND VALIDATION
OTHER ELEMENTS CHECKLIST

Notes on the use of this Checklist: Separate copies of this checklist may be completed by different individuals at different times for the various life cycle phases. Completed sections must be filled in their entirety. Unused sections may be left blank. Items identified with an asterisk (*) require that the reviewer attach his notes or describe on the reverse side of the checklist the method used or activities performed by the reviewer to verify the checklist item.

Software Item Name: _____
Software Item ID: _____

Other Elements	Yes	N/A
If required, have installation/integration tests been successfully performed and documented?	___	___
Have access controls on the program source been implemented?	___	___
Is the software identified by a unique code name and version number which appears on its hardcopy output?	___	___
Are all applicable elements of the Verification and Validation Report present?	___	___

Cover sheet
Design description, or reference
Requirements document, or reference
Test plan, or reference
Test results evaluation
Test results hardcopy computer output where available/appropriate
User documentation, or reference
Source code listing
Computer Code Certificate
Verification and Validation Checklist

EXHIBIT 4-8
CHECKLIST NO. 7 (CONTINUED)

SOFTWARE VERIFICATION AND VALIDATION
OTHER ELEMENTS CHECKLIST

Are all pages of the Software Verification and Validation Report sequentially numbered and marked with a valid Software Verification and Validation Report number?	___	___
Have all cross-outs or overstrikes in the documentation been initialed and dated by the author of the changes?	___	___

Reviewer's comments (Optional): _____

_____

Reviewed by: Name _____ Signature _____ Date _____

Exhibit 4-9
Typical Requirements Traceability Matrix

	Document A Rev. 0	Rev. 1	Document B Rev. 0	Document C Rev. 0
Requirement A	Y	Y	Y	Y
Sub Requirement 1	Y	N	Y	Y
Sub Requirement 2	N	N	Y	Y
Sub Requirement A	N	N	Y	Y
Requirement B	Y	Y	N	Y
Sub Requirement 1	Y	Y	N	Y
Requirement C	Y	Y	N	Y
Requirement D	Y	Y	Y	N

Exhibit 4-9
Typical Requirements Traceability Matrix

	Document A Rev. 0	Rev. 1	Document B Rev. 0	Document C Rev. 0
Requirement A	Y	Y	Y	Y
Sub Requirement 1	Y	N	Y	Y
Sub Requirement 2	N	N	Y	Y
Sub Requirement A	N	N	Y	Y
Requirement B	Y	Y	N	Y
Sub Requirement 1	Y	Y	N	Y
Requirement C	Y	Y	N	Y
Requirement D	Y	Y	Y	N

Exhibit 5-1
Software Change Request

SCR NUM: _____

1. Submitted by: _____ Date: ____/____/____

   Project Name: _____

2. Software Program/Document Name: _____

   Version/Revision: _____

3. SCR Type: (1-Development, 2-Problems, 3-Enhancement)

4. Short Task Descriptions

5. Detail Description:

6. Submitter's Priority [ ] 1=Critical 2=Very Important 3=Important 4=Inconvenient 5=Interesting

7. Assigned to: _____ Target Release Date_____

8. Solution Comments:

9. Software Programs affected:

10. Design Team Engineer Approval_____ Date: ___/___/___

    Design Team Group Supervisor Approval _____ Date: ___/___/___

11. Actual Release _____ Date: ___/___/___

12. Closed by:_____ Date: ___/___/___

    SCA Reference No. _____

Exhibit 5-2
Software Change Authorization

SCA Number: _____
Sheet Number: 1

Submitter:_____ System:_____ Date:__/__/__ Time__:__:__

Product Version ID:_____ Computer Name_____

Input Names	Release Names	Module Types	Software Change Request Numbers

Comments:_____

_____

_____

_____

_____

Approvals	Design Team Engineer	Design Team Group Supervisor
Signature		
Date		

EXHIBIT 5-3
SOFTWARE MODULE RELEASE HISTORY

Software Module Name: _____

Software Module Class: _____

Software Module Category: _____

RELEASE NUMBER (FF.MM.RR)	DATE	REQUESTOR	MODIFICATION	RELATED SCR NUMBER	TEST AND VERIFICATION STATUS

EXHIBIT 5-4
SOFTWARE UNIT RELEASE HISTORY

Software Unit Name: _____

Software Unit Class: _____

Software Unit Category: _____

RELEASE NUMBER (FF.MM.RR)	DATE	REQUESTOR	AFFECTED MODULE NAMES AND VERSION No.'S	SCR No. FOR THIS RELEASE	TEST AND VERIFICATION STATUS

EXHIBIT 5-5

BASELINE RELEASE HISTORY

BASELINE NAME and VERSION	DATE	CODE CERTIFICATION DATE (safety ret. only)	USE (CUSTOMERS)

EXHIBIT 6-1
COMPUTER PROGRAM ERROR NOTIFICATION

Program Name: _____ Version(s): ____

Error Number for this Program: _____

PROGRAM EXECUTABLE FILE IDENTIFICATION

	COMPUTER SYSTEM	PERMANENT FILE NAME	OWNER ID	CYCLE
1.				
2.				
3.				

ERROR DESCRIPTION:

POSSIBLE IMPACT OF ERROR:

RECOMMENDATION TO USER (ERROR AVOIDANCE AND/OR REMEDIAL ACTION):

Prepared By: _____ Date: _____

Group Supervisor: _____ CEP: _____

ABB COMBUSTION ENGINEERING NUCLEAR SYSTEMS

EXHIBIT 6-2
COMPUTER PROGRAM ERROR NOTIFICATION

RESPONSE RECEIPT

Program Name: _____ Version(s): _____

Error Number for this Program: _____

Manager: _____

I acknowledge receipt of the attached error notice, and have assessed the impact on design analyses as indicated. When the error impact is yes, appropriate actions as necessary, including 10CFR21 considerations will be taken, documented and made a quality record.

USER NAME (Print or Type)	SIGNATURE	DATE	ERROR IMPACT	
			NO	YES

Manager is to return this signed form within sixty (60) days to:

GROUP SUPERVISOR: _____ CEP: _____

ABB COMBUSTION ENGINEERING NUCLEAR SYSTEMS

EXHIBIT 7-1

Report _____ Pages
Appendices _____ Pages
Other Attachments _____ (Microfiche Sheets)

SOFTWARE VERIFICATION AND VALIDATION REPORT

Number _____ Revision _____

COMPUTER PROGRAM

Name _____ Version _____

Verification and Validation Report Prepared By: _____
Printed Name and Signature

Approved By
Group Supervisor: _____ Date _____
Printed Name and Signature

ABB Combustion Engineering Nuclear Systems
Windsor, Connecticut

EXHIBIT 7-2

COMPUTER CODE CERTIFICATE

The following code, as noted by its name, version number and executable file identification, is approved for design use.

Code Name: _____

Version Number: _____

Executable File Identification: _____

Computer(s): _____

List any limitations on use, special hardware considerations, etc.

_____
_____
_____

Verification and Validation Report Number: _____

Group Supervisor: _____

Date: _____

ABB Combustion Engineering Nuclear Systems
Windsor, Connecticut

EXHIBIT 8-1

TEST EXCEPTION REPORT

TESTER_____ TER #_____

GROUP_____ DATE_____

TEST NAME_____ SYSTEM NAME_____

STEP NO._____ SYSTEM NO._____

VERSION NO._____

VERSION DATE_____

DESCRIPTION OF EXCEPTION:

TESTER SIGNATURE_____

****

RESOLUTION

DESIGN TEAM ENGINEER_____

GROUP_____

DESCRIPTION OF EXCEPTION RESOLUTION:

DOCUMENTS CHANGED (OTHER THAN SOURCE CODE LISTINGS):

SOURCE CODE FILES CHANGED:

IMPLEMENTED BY:_____ RETESTED BY:_____

ATTACHMENT 2