GE Energy
Nuclear

NEDO-33246
Class I
eDRF# 0000-0050-2658
January 2006

LICENSING TOPICAL REPORT

ESBWR I&C
SOFTWARE INTEGRATION PLAN

INFORMATION NOTICE

This document, NEDO-33246, Rev 0, contains no proprietary information.

IMPORTANT NOTICE REGARDING CONTENTS OF THIS REPORT PLEASE READ CAREFULLY

The information contained in this document is furnished as reference to the NRC Staff for the purpose of obtaining NRC approval of the ESBWR Certification and implementation. The only undertakings of General Electric Company with respect to information in this document are contained in contracts between General Electric Company and participating utilities, and nothing contained in this document shall be construed as changing those contracts. The use of this information by anyone other than that for which it is intended is not authorized; and with respect to any unauthorized use, General Electric Company makes no representation or warranty, and assumes no liability as to the completeness, accuracy, or usefulness of the information contained in this document.

Table of Contents

Appendices

1 Introduction

This Software Integration Plan (SIntP) describes the software integration, hardware/software integration, and system integration test activities to be carried out during the development process of software based products for the ESBWR or similar projects. This Software Integration Plan (SIntP), in conjunction with other software plans, addresses and is intended to meet the requirements specified in Reg. Guide 1.170, "Software Test Documentation for Digital Computer Software used in Safety Systems of Nuclear Power Plants", and Reg. Guide 1.171, "Software Unit Testing for Digital Computer Software Used in Safety Systems of Nuclear Power Plants." This SIntP is referred as the Software Test Plan in the Man-Machine Interface System And Human Factors Engineering Implementation Plan [Ref 1.4.1 (2)].

Software Integration consists of activities that are applied during test and integration phase of the software product development. These activities encompass:

- A defined software and hardware/software integration test management approach
- A multi-tiered integration and testing strategy
- Development compliance procedures, and
- Measurement and reporting mechanisms

This Software Integration Plan (SIntP) is developed in accordance with the Software Management Plan (SMP) [Ref 1.4.2(1)] to establish procedures and guidelines necessary to prepare, execute, and document software testing for Quality Class Q software based products. Quality Class N software based products may be tested in accordance with this plan or in accordance with standard GENE procedures.

Software testing carried out in accordance with this test plan satisfies all software test requirements invoked by the Software Management Plan (SMP) [Ref 1.4.2(1)].

1.1 Purpose

The purpose of this Software Integration Plan (SIntP) is to:

- Describes the test organization defining the responsibilities of individuals or groups involved in the test and integration process,
- Describes integration and test management, such as (but not limited to) schedule, resources, security, risks and contingency planning, anomaly and problem reporting, and training needs,
- Provide a structure for software testing within the Software Coding, Integration Test and Validation Test life cycle phases,
- Provide the requirements and guidelines necessary to prepare, execute, and document software tests,
- Provide general acceptance criteria for module and integration tests,

- Define system validation test boundaries, and
- Define all required software integration and test documentation and the test deliverables, and
- Defines measurements and metrics for error tracking and resolution, and to assess the success or failure of the software integration and test effort.

1.2 Scope

The scope of this plan is to outline software integration and test strategy and techniques as required by the SMP [Ref 1.4.2(1)]. This plan also addresses software test requirements pertaining to Reg. Guide 1.170, "Software Test Documentation for Digital Computer Software used in Safety Systems of Nuclear Power Plants", and Reg. Guide 1.171, "Software Unit Testing for Digital Computer Software Used in Safety Systems of Nuclear Power Plants."

The software quality assurance requirements outlined in the Man-Machine Interface System And Human Factors Engineering Implementation Plan [Ref 1.4.1 (2)] and pertaining to software described in 10 CFR 50, Appendix B and conforming to IEEE-730-2002, IEEE Standard for Software Quality Assurance Plan are defined in the Software Quality Assurance Plan (SQAP [Ref 1.4.2(6)]).

Control of documentation (defined as configurable items) is the scope of the Software Configuration Management Plan (SCMP [Ref 1.4.2(2)]). The Software Management Plan (SMP) [Ref 1.4.2(1)] defines the scope of the design documentation requirements. This plan shall define the required software integration and test documentation.

Verification and Validation scope and requirements are defined in the Software Verification and Validation Plan (SVVP [Ref 1.4.2(3)]).

Finally, this plan shall define the software integration and test activities (both administrative and technical), the methods and tools by which to execute these activities, and the metrics by which to measure each activity. Certain test and procedures will be a subset of the SVVP [Ref 1.4.2(3)]) requirements and executed by the VVT.

1.2.1 Scope of Testing

Because of the nature of software, software tests are type tests. Testing of individual hardware units belongs under production testing which is carried out in accordance with the Quality Assurance Plan [Ref 1.4.1 (6)].

The following software test activities are covered by this plan:

1. Software Module Tests,
2. Software Integration Tests,
3. Hardware/Software Integration Tests, and

4. System Integration Tests.

This test plan shall remain in effect throughout the testing process and may be revised to reflect improved test methods, procedures and guidelines in order to achieve high functional reliability and design quality in software used in software based products for ESBWR.

Exceptions from elements in this test plan may be accepted if it is judged that the quality level of the software test item is maintained. Any deviations from this test plan must be justified and approved by the Responsible Technical Project Engineer (RTPE) and the Responsible Software Safety Engineer (RSSE). The justification and approval shall be documented in the software project Design Record File (DRF).

1.2.2 Test Boundaries

This test plan has adopted the system boundaries defined by the system codes[1] as test boundaries. However, communication links crossing system boundaries within the procurement package will be validated. This means that the following high level testing will not be carried out under this plan and is hence deferred to later system interaction tests, i.e., tests carried out under the Integrated Factory Acceptance Test (IFAT) program planned by the VVT under V&V plans.

1. Functional testing requiring inter-system communication. For instance, end-to-end validation testing involving functionality from more than one system, and

2. Validation of communication links crossing procurement package boundaries.

1.3 Acronyms, Abbreviations, and Definitions

1.3.1 Acronyms and Abbreviations

The following Acronyms and Abbreviations are used in this document:

BRT	Baseline Review Team (Part of SPE)
DRF	Design Record File
EOP	Engineering Operating Procedure
FDDR	Field Deviation Disposition Request
BRD	Build Release Description
GE	General Electric Company
GEEN	GE Energy Nuclear (Previously GENE)

[1] System codes are defined in the Project Design Manual [Ref 1.4.1 (1].

HFE	Human Factors Engineering
HSS	Hardware/Software Specification
IPS	Instrument Performance Specification
ITE	Integration Test Engineer (Part of Design Team)
MMIS	Man-Machine Interface System
MTE	Module Test Engineer (Part of Design Team)
RE	Responsible Engineer (Part of Design Team)
RETL	Responsible Engineering Technical Lead (Part of Design Team)
RSSE	Responsible Software Safety Engineer (Part of SPE)
RTPE	Responsible Technical Project Engineer (Part of Design Team)
RV	Responsible Verifier (Part of PSE)
Reg. Guide	Regulatory Guide
SCMP	Software Configuration Management Plan
SDS	Software Design Specification
SIntP	Software Integration Plan
SMP	Software Management Plan
SPE	Software Project Engineering
SPEL	SPE Lead (Part of SPE)
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SRP	Standard Review Plan
SSA	Software Safety Analysis
SSP	Software Safety Plan
SST	Software Safety Team (Part of SPE)

SVVP	Software Validation and Verification Plan
VTE	Validation Test Engineer (Part of PSE)
V&V	Verification and Validation
VVTTL	V&V Team Task Lead (Part of SPE)
VVT	Verification and Validation Team (Part of SPE)

1.3.2 Definitions

The following definitions apply throughout this document:

Acceptance Testing	- Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.
Algorithm	- A finite set of well-defined rules/mathematical equations for the solution of a problem in a finite number of steps.
Anomaly	- Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents.
Application Software Package	- A collection of software modules brought together to form a single software application, e.g., an instrument (see also System Software Package and Package).
Baseline	- A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
Baseline Review	- A formal review, conducted at the end of each process step of the software engineering design process, and requested by the Design Team's Responsible Technical Project Engineer. The baseline review process is under the control of Software Project Engineering (SPE). The Baseline Review Team (appointed by the BRT task lead engineer) performs the review. These reviews are intended to confirm adherence to the project SMP [Ref 1.4.2(1)] and SCMP [Ref 1.4.2(2)]. All Baseline Reviews are performed and documented in accordance with the Software Configuration Management Plan [Ref 1.4.2(2)], the Software Quality Assurance

		Plan [Ref 1.4.2(6)], and the Software Verification and Validation Plan [Ref 1.4.2(3)].
Branch Testing	-	Testing designed to execute each outcome of each decision point in a computer program.
Classifications of Safety-Related (Q) and Nonsafety-Related (N) Hardware and Software	-	Reference EOP 65-2.10, the Software that performs a nonsafety-related function and its common mode software failure does not defeat a safety-related function, the software is classified as Nonsafety-Related (N), (IEEE 1012 integrity level 1 and 2). The software that performs a safety-related function or its common mode software will defeat a safety-related function, it is classified as Safety-Related (Q), (IEEE 1012 integrity level 3). See Appendix A of SQAP for breakdown of software systems, which are Q class.
Code Review (Code Analysis)	-	Software source code presented to project personnel for comment or approval.
Component Testing	-	Testing conducted to verify the implementation of the design for one software element (for example, unit, module) or a collection of software elements.
Critical Software	-	Software used in a nuclear power plant safety system whose failure could have an impact on safety, or could cause large financial or social loss.
Design Record File	-	A formal controlled information record under the GEEN procedures for in-progress and completed engineering work which is retained and from which work can be retrieved.
Design Walkthrough	-	An informal review process or inspection to find defects (such as omissions, unwanted additions, and contradictions) in design documentation and to consider alternative functionality, performance objectives, or representations.
Digital System	-	For this SIntP, a digital system is an integrated system of digital hardware and associated Software (embedded or otherwise) that together, comprises a complete system or sub-system.
Documentation	-	The formal output generated by a task for a particular case (e.g., design output).

Embedded System	- A specialized computer system that is part of a larger system or machine. Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM (Firmware).
Field Deviation Disposition Request	- Field Deviation Disposition Request (FDDR) is used for documenting and disposition of the technical position for a deviation required in the field in supplied hardware, software, or services.
Firmware	- Software not running on the main processor of a computer will be considered as firmware, i.e., as software embedded in a black box device that can be used and assessed only through the device in which it is embedded.
Functional Testing	- Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions {IEEE 610.12 [2.2.2.1(1)]}.
Instrument	- A hardware device used for analytical or control functions and usually containing an embedded microprocessor(s).
Integration Testing	- An orderly progression of testing in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated.
Package	- A single source file containing one or a group of C functions, Pascal procedures or PL/M procedures and declarations, which together perform a particular job or device operation {Software Conventions and Guidelines [Ref 1.4.2(8)]).
Path Testing	- Testing designed to execute all or selected paths through a computer program.
Regression Testing	- Selective re-testing of a software item to verify that modifications have not caused unintended effects and that the software item subject to the test still complies with its specified requirements.
Responsible Engineer	- A person responsible for a given technical item (e.g., the design of an instrument).
Responsible Engineering Technical Lead	- A person with the overall responsibility for a set of DCIS software-based products. Each DCIS software-based product is assigned a RETL, including those developed by vendors.

Responsible Software Safety Engineer	- The person with overall responsibility for ensuring the safety qualities of the software including the integration of the software with the final hardware platform as defined in the SMP [Ref 1.4.2(1)].
Responsible Technical Project Engineer	- The person with overall technical responsibility for ensuring that the hardware and software design of a software-based product meets the specified requirements.
Responsible Verifier (RV)	- The person responsible for design documents verification. A responsible verifier is a qualified (i.e., by knowledge and experience) individual who did not participate in the design.
Software Feature	- A distinguishing characteristic of a software item, such as, performance, portability, or functionality.
Software Item	- Source code, object code, job control code, control data, or a collection of these items.
Software Life Cycle	- The period of time that begins when a software product is conceived and ends when the software is no longer available for use.
Software Module	- The smallest segment of code (also called routine, procedure, function or subprogram).
Software Package	- A collection of software modules (e.g., subroutines, main control tasks) brought together to form a single software product.
Software Source Code	- Computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other translator.
Software-based System	A computer system composed of specified programs that are run on commercially available hardware and/or operating systems.
Statement Testing	- Testing designed to execute each statement of a computer program.
Stress Testing	- Testing performed to evaluate a software item at or beyond the limits of its specified.
Structural Testing	- Testing that takes into account the internal mechanism of a system or component. Types include branch testing, path testing, statement testing.

System Testing	- The process of testing an integrated hardware and software system to verify that the system meets its specified requirements.
Test Case	- A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
Test Item	- Software item that is an object of testing.
Third Party Software	- Software procured from outside sources and incorporated into the ultimate software-based product.
Traceability Matrix	- A matrix that records the relationship between two or more product specifications (i.e., design documentation) of the development process (e.g., a matrix that records the relationship between the requirements and the design of a given software component).
Traceability Table	- A table that shows the relationship between functional requirements and tests. The traceability table helps ensure that all functions have been adequately tested.
Type Test	- A test that is carried out in order to evaluate the design of the test item.
Validation	- The testing process that ensures the software-based product meets its intended use and is compliant with system functional, performance and interface requirements.
Verification	- Activities performed by knowledgeable personnel independent of those responsible for the software design process to ensure that design and process outputs meet specified requirements. Design and process outputs may include software development plans, software specifications and manuals, safety analyses. Verification activities include audit, inspection, independent verification, testing, and review of operating experience.
Verification and Validation	- The verification and validation activities performed by software project engineering (SPE) in accordance with the design process (this SVVP) to ensure the quality of the associated documents produced.

1.4 Applicable Documents

Applicable documents include supporting, supplemental and reference and are given in this section. Support documents provide the input requirements to this quality plan. Supplemental documents are used in conjunction with this quality plan. Reference documents are documents that lend support, but supply no input requirements.

1.4.1 Supporting Documents

The following supporting documents were used as the controlling documents in the production of this plan. These documents form the design basis traceability for the requirements outlined in this plan.

Document Title	Document Number
1. Project Design Manual	
2. Man-Machine Interface System and Human Factors Engineering Implementation Plan	NEDO-33217
3. Quality Assurance Plan	NEDO-33181
4. Project Procurement Manual	
5. Project Management Manual	NEDC-33216
6. Composite Specification (26A6007)	A11-5299

1.4.2 Supplemental Documents

Supplemental documents are those documents that are used in conjunction with this document.

Document Title	Document Number
1. Software Management Plan	NEDO-33226
2. Software Configuration Management Plan	NEDO-33227
3. Software Verification and Validation Plan	NEDO-33228
4. Software Development Plan	NEDO-33229
5. Software Safety Plan	NEDO-33230
6. Software Quality Assurance Plan	NEDO-33245
7. Engineering Operation Procedures	NEDE-21109

Document Title	Document Number

a. 40-7.00 Design Review

b. 42-6.00 Independent Design Verification

c. 42-10.00 Design Record File

d. 55-2.00 Engineering Change Control

e. 40-3.00 Engineering Computer Programs

f. 55-3.00 Field Deviation Disposition Request

g. 45-2.00 Procurement of Engineering Services

h. 25-5.00 Work Planning and Scheduling

8. Software Conventions and Guidelines

1.4.3 Codes and Standards

The following codes and standards are applicable to the Software Integration activity to the extent specified in this plan. The applicable date/revision of the code or standard is specified in the Composite Specification [Ref 1.4.1(6)].

1.4.3.1 Electrical Power Research Institute (EPRI)

1. EPRI TR-106439, Guidelines on Evaluation and Acceptance of Commercial Grade Digital Equipment in Nuclear Safety Applications

1.4.3.2 Institute of Electrical and Electronic Engineers (IEEE)

1. IEEE 1012, Standard for Verification and Validation Plans
2. IEEE 1028, Standard for Software Reviews and Audits
3. IEEE 1228, Standard for Software Safety Plans
4. IEEE 7-4.3.2, Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations
5. IEEE 829, Standard for Software Test Documentation
6. IEEE 1008, Standard for Software Unit Testing

7. IEEE 730, Standard for Software Quality Assurance Plans

1.4.3.3 U.S. Nuclear Regulatory Commission (NRC)

1. Regulatory Guide (Reg. Guide) 1.168, Verification, Validation, Review, and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants
2. NUREG/CR-6101, Software Reliability and Safety in Nuclear Reactor Protection Systems
3. Standard Review Plan (SRP), Section 7, Branch Technical Position (BTP) HICB-14, Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems, Rev. 4
4. NUREG/CR-6463, Coding Conventions
5. 10 CFR 50, Appendix B
6. Reg. Guide 1.170, Software Test Documentation for Visual Computer Software
7. Reg. Guide 1.171, Software Unit Testing

2 Organization and Management

The software module and integration test activities are performed under the software development organization, defined in the Software Management Plan (SMP) [Ref 1.4.2(1)], called Software Design Team.

The software validation test activities are performed by V&V Team under the Software Project Engineering (SPE) organization, defined in the Software Verification and Validation Plan (SVVP) [Ref 1.4.2(3)].

The Responsible Engineering Technical Lead (RETL) has the overall responsibility for testing of the products pertaining to the DCIS procurement.

Each product (system or sub-system) in the procurement package has a Responsible Technical Project Engineer (RTPE) assigned to it. The RTPE shall establish the responsibility structure amongst members in the design team, except where independent V&V is performed.

The following sections describe the responsibilities for individuals involved in the software testing activities.

2.1 Organizational Interfaces

The test results are reported to the Responsible Engineer (RE) and Responsible Engineering Technical Lead (RETL) via various reports outlined in this document.

The Software Design Team interfaces to V&V Team under the Software Project Engineering. The Verification and Validation Team (VVT) is responsible for performing and managing V&V activities on the design and development and associated supporting documentation of the software-based product's design process to ensure that the design meets the specified requirements, to confirm the quality, safety, reliability, and performance of the design, and to ensure that the products meets its intended use.

Any interface that the Software Design Team would have with vendor organizations that provide software products to the project design organization but are not part of the software design organization are through the software design organization. The generic categories describe organizations that supply input to either software design organization or other organizations contracted by the design organization to provide part of the design function. For independent vendors, the SPE's BRT and SQA personnel shall audit that organization for compliance with the intent of the SQAP [Ref 1.4.2(6)], the SMP [Ref 1.4.2(1)], and all other applicable procedures.

2.2 Management

The Responsible Engineering Technical Lead (RETL) has management responsibility for the software testing of the set of DCIS software-based products assigned to him or her. The management tasks described below may be delegated to persons within the project or vendor organization but the management responsibility remains with the RETL.

2.2.1 Scheduling and Planning

The RTPE has the overall responsibility for scheduling and planning all software integration and test tasks and activities. Each Test Team (RE, Module Testers, Integration Test Engineers, Validation Test Designers, and Validation Testers) is responsible for the management, scheduling, and planning activities for their respective sub-organization.

The schedule for software testing activities shall be integrated in the detailed Work Plans developed for each work package as required by Section 6.1 of the Software Development Plan [Ref 1.4.2(4)].

2.2.2 Resources

Resource management includes the following elements:

1. Determination of the resources needed. Resources include:
    a. Qualified test engineers,
    b. Test facilities,
    c. Test equipment and tools, and
    d. Special procedures if required, for such as, security and access control.
2. Coordination of these resources for each level of software testing.

2.2.3 Training Needs

The training needs shall be assessed and necessary training shall be carried out to assure the adequacy of the software testing process.

2.2.4 Reviews

The progress of testing and issues related to testing shall be evaluated on a regular basis, e.g., in periodic staff meetings. Special attention shall be paid to circumstances that indicate deficiencies in the testing process. Necessary corrective actions must be taken to improve the test process if a deficiency is identified. This test plan shall be updated accordingly.

2.2.5 Contingency Planning

It shall be ensured to the extent practical that staffing plans can be reinforced if required. This may require agreements in advance with other project teams or other organizations to acquire individuals for temporary project assignment.

2.3 Roles and Responsibilities

2.3.1 Test Team

2.3.1.1 The Responsible Technical Project Engineer (RTPE)

The RTPE, as part of the Design Team, has technical responsibility for the software testing tasks related to the product he or she is assigned responsibility. This implies responsibility for coordination of all levels of software testing activities related to the product.

2.3.1.2 Responsible Engineer (RE)

The RE, within the Design Team, is responsible for tasks related to testing of the software item he or she is assigned responsibility.

2.3.1.3 Software Module Tester

The Software Module Tester, within the Design Team, is responsible for designing, executing and documenting the portion of the module test he or she is assigned.

2.3.1.4 Integration Test Engineer

The Integration Test Engineer, within the Design Team, is responsible for designing, executing and documenting the integration test.

2.3.1.5 Validation Test Designer

The Component Validation Test Designer and the System Validation Test Designer, within the SPE, are responsible for designing the component validation test and the system validation test respectively.

2.3.1.6 Validation Tester

The Validation Tester, within the SPE, is responsible for executing and documenting the portion of the validation test he or she has been assigned.

2.3.2 Software Integration and Test Personnel Qualifications

The test team shall write and/or execute, in whole or portions, of the Integration and Module testing. As such team members shall be required to know the system, design, and implementation techniques (including tools) employed by the design organization.

3 Procedures

The software integration phase tasks include Module Test and Integration Test of new developed software as well as modifications to any previously developed software. The Hardware/Software Integration and System Integration tests are applied to all the software including Third Party/Vendor Software, unmodified Previously Developed Software, and Commercial off the Shelf Software (COTS). This section outlines strategy and guidelines for Software Integration, Hardware/Software Integration, and System Integration tests.

3.1 Software Classifications

3.1.1 Third Party/Vendor Software

Third Party/Vendor software can include libraries, functions, or vendor applications developed expressly for the ESBWR project. The software design team prepares evaluation reports:

1. Evaluating technical, quality, and security requirements, and identifying critical characteristics of commercial digital equipment to be used in safety systems

2. Identifying critical characteristics, accepting digital products from commercial vendors, and dedicating them for use in nuclear safety-related applications

3. Ensuring the dedication and security basis is maintained to and remains valid over the operating lifecycle of the equipment

3.1.2 Previously Developed Software

Previously developed software is software developed by GEEN prior to the ESBWR project. The software design team prepares PDS evaluation reports:

1. Ensuring that any supplemental development or modification to the previously developed software for the ESBWR project were done in accordance to applicable plans, procedures, or processes

2. Stating that appropriate methods were identified, and instituted by the design team (e.g., auditing, inspection or independent review) for verifying that original design remains intact and that critical characteristics of the software have been properly considered for use in digital products, from commercial vendors or from GEEN, that will be dedicated for use in nuclear safety-related applications

3.1.3 Commercial off the Shelf Software (COTS)

Commercial off-the-shelf software can include libraries, functions, or complete applications that are commercially available to anyone and are used in the software-based applications. The software design team prepares COTS reports:

1. Determining technical and quality requirements, and identifying critical characteristics of commercial digital equipment to be used in safety-related systems
2. Identifying critical characteristics, accepting digital products from commercial vendors, and dedicating them for use in nuclear safety-related applications
3. Ensuring the dedication and security basis is maintained to and remains valid over the operating lifecycle of the equipment

The examination shall be based on EPRI-TR-106439, Guidelines on Evaluation and Acceptance of Commercial Grade Digital Equipment in Nuclear Safety Applications [Ref 1.4.3.1 (1)].

3.1.4 Support Software/Tool Acceptance

The design team prepares support software tool reports for all support software:

1. Evaluating that the tool is acceptable for the safety classification of the software it is to be applied to
2. Submitting such reports exists from prior projects

3.2 General Software Test Guidelines

The following is the set of general test tasks to be followed at all levels of testing, from module tests to validation (system integration) tests:

1. Test preparation,
2. Test design
3. Test execution, and
4. Test summary.

Tasks that are specific to a particular test are described in its respective section.

3.2.1 Test Preparation Guidelines

The purpose of test preparation is to assure that the required test activities can be properly carried out within the project schedule. This is accomplished by identification of resources required to support the development, execution, and the documentation of the test. The individual responsible for test preparation shall carry out the following tasks:

1. Define the scope of the test and identify the software items to be tested,

2. Design a detailed test schedule aligned with the project plan,
3. Specify test prerequisites,
4. Specify the test environment,
5. Identify equipment, documentation, tools and instrumentation needed for the accomplishment of the test,
6. Assign qualified[2] test designers[3](s) and tester(s),
7. Ensure the training needs are satisfied, and
8. Initiate the creation of the Test Report.

3.2.2 Test Design Guidelines

The purpose of this section is to expand on the test approach described in the SMP [Ref 1.4.2(1)], to help assure completeness of the test. The test designer shall perform the following tasks, where applicable:

1. Specify the software features to be tested for each software item,
2. Specify the software features not to be tested and justify why it will not to be tested (e.g., previously tested unmodified feature),
3. Determine the test approach and specify the test techniques to be used,
4. Specify the test cases,
5. Develop the test procedures and instructions, and
6. Define the pass criteria.

The following software test elements shall be considered when the test approach is determined:

1. White-box testing. White-box testing, also known as structural testing, is a test methodology in which test steps are based on knowledge of the internal structure of the software module, or a group of software modules. It may execute all the statements or branches in the software module to check how the system is implemented. Methods to be used for white-box testing include:

   a. Branch testing: A testing technique to execute each outcome of each decision point in a computer program.

[2] See section SVVP [1.4.2(3)] Section 2.2.1 for qualification requirements.

[3] Software Module Tester(s), Integration Test Engineer(s) and Validation Test Designers(s) depending on type of test (see Section 2.3.1).

b. Path testing: A testing technique design to exercise every independent execution paths through the computer program.

c. Statement testing: A testing technique design to execute each statement of a computer program.

2. Black-box testing. Black-box testing is a test methodology that uses requirements external to a feature to derive test cases and test procedures. It verifies the end results at the feature I/O level, but does not check on how the feature is realized, nor does it assume that all statements related to the feature are executed. Methods to be used for black-box testing include:

a. Module interface testing: Testing performed to evaluate whether the values along the interface are correct as they relate to software modules that call them.

b. Interface testing: Testing performed to detect errors that may have been introduced into the system because misinterpretation of the interface specification.

c. Regression testing: Selective re-testing of a software item to verify that modifications have not caused unintended effects and that the software item subject to the test still complies with its specified requirements

d. Stress testing: Testing performed to evaluate a software item at or beyond the limits of its specified requirements.

Reviews in the form of design walkthroughs described in the SVVP [Ref 1.4.2(3)], shall be carried out during the test design process to evaluate the adequacy of the selected test strategy and to assure that all test features are identified.

3.2.3 Test Execution Guidelines

The purpose of test execution is to expose the software test item to conditions that may reveal potential implementation errors. Test execution includes the following tasks:

1. Obtain test items including relevant reference documentation,
2. Set the test environment,
3. Document the test results while executing the test procedures, and

4. Initiate the change control process[4] to resolve design errors encountered during the test.

3.2.4 Test Summary Guidelines

The purpose of the test summary is to facilitate review and evaluation of adequacy of the test. The test summary shall include:

1. The test activities,
2. The test results,
3. V&V traceability, and
4. All test incidents and the associated resolutions.

The purpose of the V&V traceability summary is to demonstrate that every functional requirement, performance requirement, and user interface requirement in a specification has been verified by the test. A table with all requirements listed along references to appropriate sections in the test report is an acceptable method to satisfy this purpose.

4 Methods

Methods, tools, software, and hardware used to perform software integration tests are described in the following sections.

4.1 Software Testing

4.1.1 Module Testing

Module testing is verification of the internal structure of individual software modules, or a group of modules, to ensure that each software function allocated in the Software Design Specification (SDS) performs as intended.

Module test preparation must be completed before any other module test activity may commence. Module test design, execution and summary however, may be carried out in an incremental manner. In other words, a test step in a test procedure for a certain test case may be executed and documented before the next test step or test case is defined.

For Quality Class Q software, module testing shall not be considered to be a primary element in the debugging process. A Quality Class Q software module shall be debugged, code reviewed and supposedly error free before it is subject to module testing.

Module tests for Quality Class Q functional software can be carried out using a debugger (e.g., in-circuit-emulator) attached to an instrument with needed hardware and software modules installed and operating. Module tests carried out in such an environment overlap integration tests since errors associated with integration issues may be detected during the module test, which represents an important echelon in the defense in depth approach used for Quality Q software.

[4] The change control process is described in Section 3.5.1 of the SCMP [Ref 1.4.2(2)],

4.1.1.1 Module Test Preparation

The RE shall prepare the module test in accordance with the test preparation guidelines in Section 3.2.1.

The software items to be tested are:

1. New software modules,
2. Modified software modules,
3. Modules required to be tested as a result of a PDS Evaluation, and
4. Third party software modules.

If Quality Class N software is to be tested, the RE shall define the combinations of modules that form the test items and also decide on the depth of the testing to be performed.

All performed test preparation activities shall be documented in the Module Test Report as described in Section 4.2.1.2.

4.1.1.2 Module Test Design

The software module tester shall design the module test in accordance with the test design guidelines in Section 3.2.2.

It is within the Module Test designer's responsibility to:

1. Identify the reference documentation to each test item, such as:
    a. Software Design Specification (SDS),
    b. Data Communication Protocol Specifications, and
    c. For previously developed software:
        i) Associated DRF(s) and module test report(s) and
        ii) Relevant problem reports (e.g., NUMAC Problem Reports, Field Deviation Disposition Requests, service requests from customer(s) due to equipment failure or the result of non-conformance) and the associated resolution reports.

Quality Class N Software

The software module test designer shall use the RE's directives from the module test preparation regarding features to be tested and depth of testing to:

1. Identify the functions to be tested,
2. Define test cases necessary to achieve the required coverage,

3. Determine the appropriate application of white-box and black-box testing techniques to achieve the required depth of testing, and

4. Define pass criterion for each test case.

The results of the test design activities performed shall be documented in the Module Test Report.

Quality Class Q Software

To assure the completeness of the test, i.e., that all software features and combinations of software features and associated procedures, states, state transitions, and associated data characteristics essential to the safety are included in the test, the module test designer shall:

1. Specify the features to be tested. The test features shall include:

   a. All module interfaces,

   b. All local data structures,

   c. All unconditional processing paths,

   d. All conditional processing paths, and

   e. All loops.

2. Specify test cases and test procedures using the white-box testing techniques or combinations of white-box testing and black-box testing techniques such that:

   a. The module interfaces are tested to ensure that information is properly exchanged with the module under normal and out of range (abnormal) conditions,

   b. Every local data structure is validated and examined to ensure that its integrity is maintained during an algorithm's entire life cycle,

   c. All processing paths through the control structure are exercised to ensure that all statements in a module have been executed at least once,

   d. The boundary conditions are tested to ensure that the module operates properly at boundaries established to limit data or restrict processing,

   e. All security functions are tested to ensure that the software is secure and it protects against undesired accesses,

   f. Each loop is tested at its boundaries and within its operational bounds.

   g. A diagnostic or error detection feature is tested with faults(s) or error(s) present to the extent feasible to ensure proper detection and handling,

h. Processing time is measured, or verified against limits, for time critical software modules, and

i. An algorithm used for the first time, and fully encompassed by the test item, is validated unless it is validated elsewhere.

3. Specify pass criterion for each test case.

4. Ensure that every functional requirement in the SDS will be tested and that the test procedure allows the test results to be documented in a traceable[5] manner.

The module tester shall document the test design in the Module Test Data Sheet.

4.1.1.3 Module Test Execution

The software module tester shall execute the module test in accordance with the test execution guidelines in Section 3.2.3 using the test procedures and test instructions developed during the module test design.

The module tester shall document the result of the execution of each test step in the Module Test Data Sheet (described in Section 4.2.1.1) and determine if the software feature passed or failed based on the defined pass criteria.

If the test reveals an error in module design or implementation, the module tester shall document it in the Module Test Data Sheet.

Regression testing may be performed on modified software modules if it is justified that such testing does maintain the integrity of the software item. The software module tester shall document the justification in the Module Test Data Sheet.

4.1.1.4 Module Test Summary

The module tester shall summarize all module test activities in accordance with the guidelines in Section 3.2.4.

The summary shall be documented in the Module Test Report as described in Section 4.2.1.2 below.

4.1.2 Integration Testing

Integration testing is an orderly progression of testing to uncover errors associated with software and hardware interfaces. Integration testing is performed to verify that all software modules perform as intended after being installed in the instrument and that the instrument conforms to the requirements in the IPS and the User's Manual.

Integration test preparation must be completed before any other integration test activity may commence. Integration test design, execution and summary however, may be carried out in an incremental manner. In other words, a test step in a test procedure for a certain test case may be executed and documented before the next test step or test case is defined.

[5] Traceable manner is required to facilitate the V&V traceability analysis to be performed in the module test summary.

Some of the integration test objectives are similar to the validation test objectives (hardware/software integration) described in Section 4.1.3.2.1 below. The difference is in the way these tests are performed. An engineer familiar with the detailed design using white-box and combinations of white-box and black-box testing techniques shall carry out integration testing, while component validation shall be carried out by an individual less involved in the design using black-box techniques only.

Like the overlap between module testing and integration testing, the overlap between integration testing and validation testing represents an important echelon in the defense in depth approach.

4.1.2.1 Integration Test Preparation

The RE shall prepare the integration test in accordance with the test preparation guidelines in Section 3.2.1.

The software item to be tested is:

1. Baseline source code from the coding phase

The RE shall specify whether the integration test shall be carried out in an incremental or non-incremental manner. Incremental testing involves testing a small part of the software package and then incrementing the configuration of the software package under test by adding one component at a time and testing after each increment.

Non-incremental testing involves assembling all components of the software package and testing them all at once.

There are two (2) ways of carrying out incremental testing.

- Top-down testing: Top-down testing is a process of integrating the Application Software Package under test, progressively, from the top (i.e., main control module) to the bottom (i.e., lower level software modules) during testing to complete the software package.
- Bottom-up testing: Bottom-up testing is the process of incrementing the Application Software Package under test, progressively, from the bottom to the top.

Selection of the test method depends on the characteristic of the software and the quality assurance requirement classification (Quality Class Q or N). If Quality Class N software is to be tested then the RE shall also provide guidelines for the integration test designer how to achieve a depth of testing commensurate with the depth of testing aimed at during the corresponding module test. The selected test method and possible guidelines shall be documented in the Integration Test Report.

The results of the test preparation activities performed shall be documented in the Integration Test Report (described in Section 4.2.2.4).

4.1.2.2 Integration Test Design

The Integration Test Engineer shall design the integration test in accordance with the test design guidelines in Section 3.2.2. The following specific instructions shall be followed at integration test design for Quality Class Q software. For Quality Class N software, however, these instructions may be modified by the guidelines provided by the RE (see Integration Test Preparation above).

To assure the completeness of the integration test, the Integration Test Engineer shall:

1. Observe the test method and guidelines specified by the RE during the integration test preparation.

2. Specify system build method (Section 4.2.2.1 provides details),

3. Identify the entities to be tested. The test entities include:

   a. S/W architecture: It shall be verified that the operating system controls the application as specified and that the tasks interact properly.

   b. Interfaces: The integration test shall be designed to uncover the following interface related errors:

      i) Module I/O arguments mismatch

      ii) Improper global data access,

      iii) Stack mismanagement,

      iv) Incorrect interrupt or exception handling, and

      v) Incorrect external information interchange (through hardwired I/O, data links and user interface).

   c. Access: Testing shall be performed to verify the protection against unauthorized and unintended access to the system. Security tests are performed to help ensure that the software does not allow unauthorized and unintended access.

4. Specify test cases and test procedures using combinations of white-box and black-box testing techniques. Each feature shall be evaluated for test relative

a. Range: Interfaces shall be range tested to ensure that values are properly passed through the instrument's inputs and outputs over the entire process range and that values are limited at the range boundaries as specified.

b. Performance: The feature shall be tested to ensure that speed and accuracy requirements are met.

c. Stress: Stress testing shall be performed to ensure that the instrument operates under abnormal conditions.

d. Security: Testing shall be performed to verify the protection against unauthorized and unintended modification of user settable parameters is implemented as specified. Security tests are performed to help ensure the robustness of the instrument.

e. Screen: Screen testing shall be performed to help ensure the user interface is implemented correctly. As a minimum, it shall include all functionality necessary to navigate trough the screens under all possible conditions.

f. Help system: Testing shall be performed to ensure that the help system generates the appropriate messages when help is sought from all possible approaches.

g. Error messages: Every condition that will initiate the software to generate an error message shall be identified. Testing shall be performed to ensure the appropriateness and understandability these messages.

h. Self-supervision: The self-supervision functions shall be tested to the extent feasible to ensure proper processing upon detection of faults.

5. Specify pass criteria based on the requirements specified in the Instrument Performance Specification (IPS) and the User's Manual, and

6. Ensure that every functional and performance requirement in the IPS, User's Manual, and the SDS[6] will be tested and that the test procedure allows the test results to be documented in a traceable[7] manner.

[6] Requirements that were tested during the module test and are listed in the V&V traceability table in the Module Test Report do not need to be re-tested.

The Integration Test Engineer shall document the test design in the Integration Test Data Sheet.

4.1.2.3 Integration Test Execution

The Integration Test Engineer shall execute the integration test in accordance with the test execution guidelines in Section 3.2.3 using the test procedures and test instructions developed during the integration test design.

The Integration Test Engineer shall document the execution of each test step in the Integration Test Data Sheet (described in Section 4.2.2.2) and determine if the software feature passed or failed based on the defined pass criteria.

If the test reveals an error, the Integration Test Engineer shall document it in the Integration Test Data Sheet and add a tally mark in the appropriate section of the errors tally sheet (described in Section 4.2.2.3). The tally marking may be postponed to the Integration Test Summary.

Regression test may be performed on modified software modules if it is justified that such testing does maintain the integrity of the software item. The Integration Test Engineer shall document the justification in the Integration Test Data Sheet.

4.1.2.4 Integration Test Summary

The Integration Test Engineer shall summarize the integration test activities in accordance with the guidelines in Section 3.2.4 and document the result in the Integration Test Report. The Integration Test Engineer shall also attach the error tally sheet to the Integration Test Report.

4.1.3 Validation Testing

Validation Testing relies entirely on black-box testing techniques and is executed using formally prepared test procedures. The purpose of validation testing is to demonstrate that:

1. Each individual instrument conforms to all functional and performance requirements specified in the IPS. This is referred to as instrument level validation, and

2. The software-based product is operational and conforms to all functional and performance requirements specified in the HSS. This is referred to as system level validation.

Validation testing should be carried out on production units. If this is not feasible, prototype units may be used with subsequent verification of equivalency between those and the production units.

4.1.3.1 Validation Test Preparation

The RE's and the RTPE shall follow the guidelines in Section 3.2.1 to prepare the instrument level validation tests and system validation test respectively.

The software items subject to validation testing are:

1. Individual baseline integrated Application Software Packages for all software-based instruments in the system.

---

[7] Traceable manner is required to facilitate the traceability analysis to be performed in the integration test summary.

2. Baseline integrated System Software Package, i.e., Application Software Packages for all software-based instruments combined into a software-based product.

In addition to the test preparation tasks to be performed, RTPE shall also coordinate instrument level and system level validation testing.

The RE's and the RTPE shall also initiate the development of the Validation Test Report for each instrument and software-based product respectively.

4.1.3.2 Validation Test Design

The Instrument Level Validation Test Designers and the System Level Validation Test Designer are responsible for the development of the Validation Test Procedures and Test Cases Specifications (see Section 4.2.3.1 for details) for the instruments and the software-based product respectively.

A Validation Test Procedures and Test Cases Specification is a formal test document that has to be reviewed and approved in advance of the test.

4.1.3.2.1 Instrument Level Validation Test Design

The Instrument Level Validation Test Designer shall design the instrument validation test in accordance with the test design guidelines in Section 4.1.2.2.

In addition to this the Instrument Level Validation Test Designer shall ensure that the Instrument Level Validation Test Procedures and Test Cases Specification is designed such that:

1. It can be demonstrated that the instrument meets all functional and performance requirements in the IPS,

2. It can be demonstrated that the instrument performs consistently with the information in the User's Manual, and

3. Ensure that the test procedure allows the test results to be documented in a traceable manner.

4.1.3.2.2 System Level Validation Test Design

The System Level Validation Designer shall design the system validation test in accordance with the test design guidelines in Section 4.1.2.2.

In addition to this the System Level Validation Test Designer shall:

1. Specify the configuration of the system, and

2. Ensure that the System Level Validation Test Procedures and Test Cases Specification is designed such that:

    a. It can be demonstrated that the system meets all functional and performance requirements in the HSS,

b. It can be demonstrated that the system performs consistently with the information in the User's Manuals, and

c. Ensure that the test procedure allows the test results to be documented in a traceable manner.

4.1.3.3 Validation Test Execution

The validation tester(s) shall execute the validation test accordance with the procedures defined in the Validation Test Procedures and Test Cases Specification (see Section 4.2.3.1).

The validation tester(s) shall create an anomaly list and a detailed anomaly report if a deviation from a specification is found during validation testing.

The test results shall be documented in the space provided in the Validation Test Procedures and Test Cases Specification and any generated anomaly documentation shall be attached to the Validation Test Report (see Section 4.2.3.2).

4.1.3.4 Validation Test Summary

The RE and the RTPE shall summarize validation test activities, for component and system validation respectively, in accordance with the guidelines in Section 3.2.4. The summary shall be documented in the Validation Test Report

4.2 Documentation and Problem Reporting

This section specifies requirements for the documentation generated during the testing governed by this plan. Additional information about software test documentation may be found in IEEE 829[1.4.3.2(5)].

Please refer to the SVVP [1.4.2(3)] for document verification requirements and the SCMP [1.4.2 (2)] for document repository.

4.2.1 Module Testing

4.2.1.1 Module Test Data Sheet

Test results for each software module shall be documented on individual Module Test Data Sheets.

For Quality Class N software, the test results for each software item shall be documented on individual Module Test Data Sheets to a depth at the discretion of the module tester. However, the results must be documented in sufficient detail to allow the reviewer to judge the adequacy of the test.

A sample form of a Module Test Data Sheet is shown in Appendix B.1. It is not required that this format is followed, provided that the following information is included:

1. The identity of the software module tester with signature and date fields,

2. Test item identification:

a. Module name including revision level,

b. Software package name including revision level, and

c. Quality Assurance requirement classification (i.e., Quality Class Q or N).

3. Test design, execution activities, and results (described in Section 4.1.1.3)

a. Test cases and pass criteria,

b. Test procedures and test instructions,

c. For Quality Class Q modules: the execution of each test step,

d. For Quality Class N items: a summary of the test execution,

e. Findings,

f. Justification of regression testing if such testing is used, and

g. Any incidents pertinent to the test.

All Module Test Data Sheets for the module test shall be attached to the Module Test Report.

4.2.1.2 Module Test Report

The Module Test Report shall document the test activities and test summary to demonstrate that the required module testing activities have been completed. A Module Test Report shall include the following information:

1. Module Test Preparation (details in Section 4.1.1.1),

a. Scope of the module test,

b. Identity of the individual responsible for the test preparation (name, signature and date),

c. Detailed module test schedule,

d. Test prerequisites,

e. Equipment, tools and instrumentation used to accomplish the testing, and

f. Module test guidelines and test item definitions (for Quality Class N)

2. Module Test Design (details in Section 4.1.1.2),

a. Software features subject to test,

b. Software features not subject to test,

c. Test approach and test techniques

d. Test cases, test procedures and test instructions, and

e. The pass criteria

Item 2c, 2d, and 2e may be references to the module test data sheets below.

3. Module Test Execution (details Section 4.1.1.3),

   a. The individual Module Test Data sheets (Section 4.2.1.1),

   b. Testing logs if used.

4. Module Test Summary (details in Section 4.1.1.4),

   a. Code review data sheets (see SMP [Ref 1.4.2(1)]]).

   b. Summary of the findings and their resolutions.

   c. V&V traceability analysis

The Module Test Report shall also include the following information:

1. The identity of the individual responsible for approving this report (name, signature and date),

2. A verification statement signed and dated by the verifier,

3. A reference to this test plan.

4.2.2 Integration Testing

4.2.2.1 Preliminary Build Release Description (BRD)

A preliminary[8] Build Release Description (BRD) shall be prepared for each instrument to describe the actual construction of the Application Software Package from packages and/or libraries. The preliminary BRD shall be tailored to the particular development system used. The BRD shall include the following information:

1. Identity of the Integration Test Engineer

2. Identity of instrument and the project name,

[8] The final Build Release Description will be issued together with the software at completion of the Validation Test phase.

3. All required software programs, including source files, command files, data and library files, that are part of the build and their revision level,
4. The requirement specification documents (i.e., IPS and User's Manual),
5. Procedure to build the software or make file name,
6. The location in the configuration management system where the identified files are stored,
7. Tools used in the build activity, and
8. A verification statement signed and dated by the verifier.

The BRD shall be attached to the Integration Test Report.

4.2.2.2 Integration Test Data Sheet

Individual Integration Test Data Sheets shall be prepared for each feature tested.

For Quality Class N software, the depth of documentation is at the discretion of the Integration Test Engineer. However, the results must be documented in sufficient detail to allow the reviewer to judge the adequacy of test execution.

A sample form of an Integration Test Data Sheet is shown in Appendix B.2. It is not required that this format is followed, provided that the following information is included:

1. The identity of the Integration Test Engineer with signature and date fields,
2. Test feature identification:
   a. Name or a description of the feature
   b. Software package name including revision level, and
   c. Quality Assurance requirement classification (i.e., Quality Class Q or N).
3. Test design, execution activities, and results (described in Section 4.1.2.3)
   a. Test cases and pass criteria,
   b. Test procedures and test instructions,
   c. For Quality Class Q features: the execution of each test step,
   d. For Quality Class N features: a summary of the test execution,
   e. Findings,
   f. Justification of regression testing if such testing is used, and

g. Any incidents pertinent to the test.

The Integration Test Data Sheet shall be included as part of the Integration Test Report.

4.2.2.3 Errors Tally Sheet

The errors tally sheet provides means for detecting significant deficiencies in the code development process. This sheet shall be prepared for each integration test and may be filled out during the integration test execution or at the integration test summary.

The following classes of errors have been defined and shall be counted to facilitate evaluation of the performance of the code development process.

Declaration, definition, and resources reference errors:	Conflicts between actual use and intended use of (e.g., constants, variables, functions etc.).
Control flow errors:	Errors in the inter-module calling structure.
Computational errors:	Errors causing analysis faults.
Task interaction errors:	Errors in software architecture implementation.
Other errors:	Errors, which are not in the classes above.

A sample form of an Errors Tally Sheet is shown in Appendix B.3. It is not required that this format is followed, provided that the error types above are included.

4.2.2.4 Integration Test Report

The Integration Test Report shall document the test activities and test summary to demonstrate that the required integration testing activities have been completed.

An Integration Test Report shall include the following information:

1. Integration Test Preparation (details in Section 4.1.2.1),
   a. Scope of the integration test,
   b. Identity of the individual responsible for the test preparation (name, signature and date),
   c. Detailed integration test schedule,
   d. Test prerequisites,
   e. Equipment, tools and instrumentation used to accomplish the testing, and
   f. Integration test guidelines
2. Integration Test design (details in Section 4.1.2.2),

a. Software features subject to test,

b. Software features not subject to test,

c. Test approach and test techniques

d. Test cases, test procedures and test instructions, and

e. The pass criteria

f. Preliminary Build Release Description (Section 4.2.2.1), and

Item 2c, 2d, and 2e may be references to the integration test data sheets below.

3. Test execution (details Section 4.1.2.3),

   a. The individual Integration Test Data sheets (Section 4.2.2.2),

   b. Testing logs if used.

4. Test summary (details in Section 4.1.2.4),

   a. Summary of the findings and their resolutions,

   b. The errors tally sheet (Section 4.2.2.3)

   c. V&V traceability analysis

The Integration Test Report shall be also include the following information:

1. The identity of the individual responsible for approving this report (name, signature and date),

2. An independent verifier's statement of conformance signed and dated,

3. A reference to this test plan.

4.2.3 Validation Testing

4.2.3.1 Validation Test Procedures and Test Cases Specification

The Validation Test Procedures and Test Cases Specification shall include the following information:

1. Description of the purpose of each test case and the associated procedure(s),

2. Methods for logging observations.

3. Description of each test procedure in detail.

Each procedure shall contain information about:

a. What to document before the execution of a test case, e.g., date, time of day, responsible tester etc.

b. How to set up the equipment and the environment (test configuration) for the test case,

c. Actions necessary to deal with anomalous events that may occur during test execution.

d. Prerequisites and actions for each test step,

   i) Conditions to be met before proceeding with the test step

   ii) How to make a measurement (e.g., response time) if required,

   iii) What to document.

Each procedure may also contain information about:

a. Steps necessary to suspend (shut down) testing in the middle of a procedure,

b. Identify possible restart points after a shut down and describe the actions required to restart the procedure at each of these points.

The Validation Test Procedures and Test Cases Specification shall provide space for documentation of the test result for each test step and a reference to an anomaly report.

4.2.3.2 Validation Test Report

The Validation Test Report shall document the test activities and the test summary to demonstrate that the required validation testing activities have been completed.

A Validation Test Report Test Report shall include the following information:

1. Test Preparation (Section 4.1.3.1),

   a. Scope of the validation test,

   b. Detailed validation test schedule,

   c. Test Environment, such as system configuration,

   d. Equipment, tools and instrumentation used to accomplish the testing,

   e. Identity of the responsible test designer (name, signature and date), and

   f. Identity of the responsible tester (name, signature and date).

2. Test design (Section 4.1.3.2),

    a. A reference to the Validation Test Procedures and Test Cases Specification (details in Section 4.2.3.1).

3. Test execution (Section 4.1.3.3),

    a. The Validation Test results (a reference to the Validation Test Procedures and Test Cases Specification),

    b. Testing logs if used.

4. Test summary (Section 4.1.3.4),

    a. Summary of test activities,

    b. Summary of the test results, including all resolved incidents and their resolutions,

    c. Traceability analysis (e.g., traceability table).

5. The identity of the individual responsible for approving this report (name, signature and date),

6. A verification statement signed and dated by the verifier, and

7. A reference to this test plan.

4.3 Approvals

All test documentation produced must be approved by the RETL.

4.4 Test Deliverables

The following items constitute the test deliverables:

1. Module Test Reports,

2. Integration Test Reports,

3. Validation Test Procedures and Test Cases Specifications,

4. Validation Test Reports, and

5. Test code and data supporting the above reports.

4.5 Measurement & Use of Metrics

The test coverage measurements and metrics should be designed to measure performance of software in both abnormal conditions as well as mishaps. These measurements and metrics are defined in the Software Verification and Validation Plan (SVVP [Ref 1.4.2(3)]).

4.6 Acceptable Defect Levels

Acceptable defect level targets shall be established for various phases of software integration. The number of acceptable defects should be lower for Quality Class Q software compared to Quality Class N software. A significant effort shall be expanded on Quality Class Q software and this effort shall not be diluted on nonsafety-related (Quality Class N) software. The acceptable defect level targets should be lower in each subsequent software integration phase. The number of acceptable defects should be lower during integration test phase compared to module test phase. The number of acceptable defects should be lower during validation test phase compared to integration test phase and should practically reach zero. If the number of defects is not lower in each subsequent phase, the software development process should be re-examined to find the causes for poor software quality and adjusted to fix the problems.

Appendix A: Figures

A.1 Software Testing Responsibility Structure

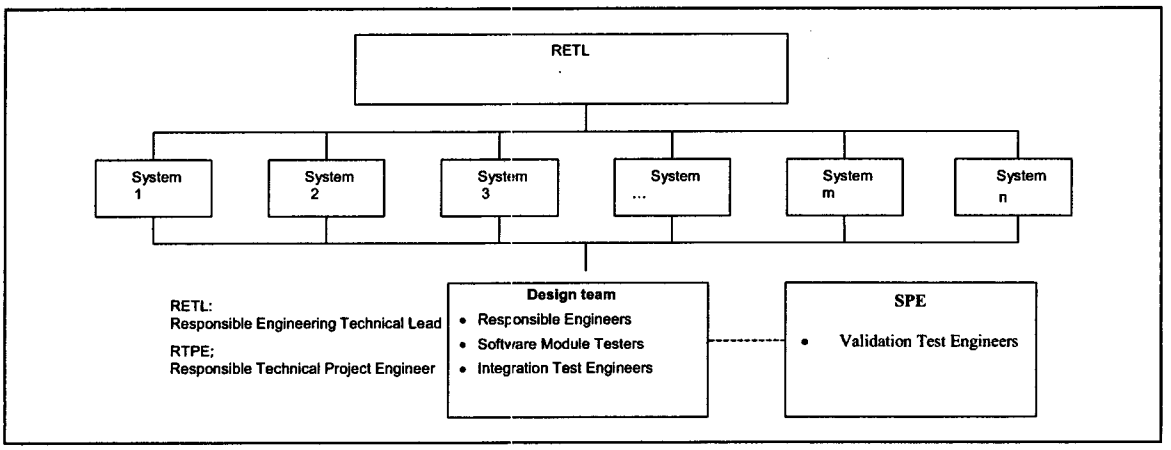

Figure 4-1: Software Testing Responsibility Structure

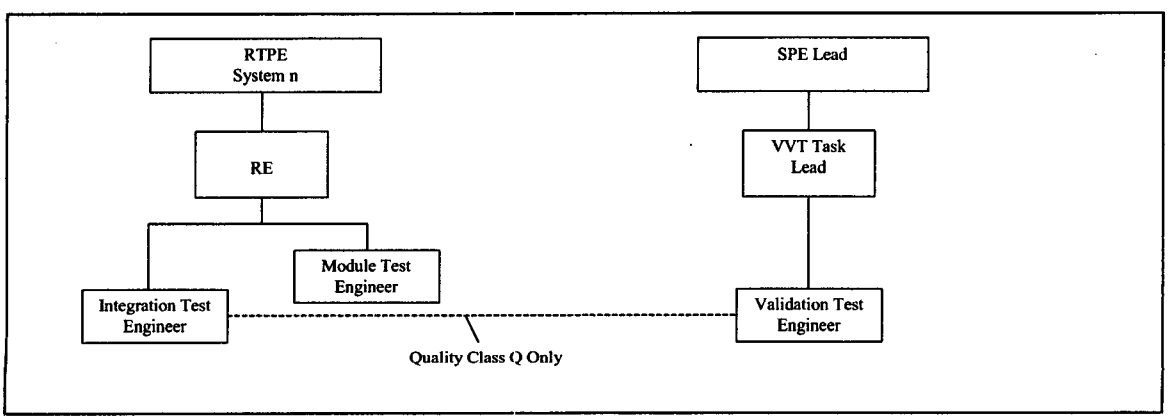

Figure 4-2: Software Product Related Testing Responsibility Structure

Appendix B: Report Format

B.1 Module Test Data Sheet (example)

MODULE TEST DATA SHEET DRF# _____

Page: _____ of _____

Responsible Engineer:		**Page:** of
Module Tester:	**Sign:**	**Date:**

Package:	**Quality Class:**	**Rev:**
Module:		**Rev:**
Storage Location:	**Module Designer:**	

Test Description	Finding	Status

B.2 Integration Test Data Sheet (example)

INTEGRATION TEST DATA SHEET DRF# _____

Page: _____of _____

Responsible Engineer:			**Page:** of
Integration Test Engineer:	**Sign:**		**Date:**
Application S/W Package:		**Rev:**	**Quality Class:**
Location:		**Instrument:**	
Test Entity:			

Test Description	**Finding**	**Status**

B.3 Errors Tally Sheet (example)

INTEGRATION ERRORS TALLY SHEET DRF# _____

Page: 1 of 1

Application Software Package: Revision:

Instrument: Quality Class:

Total Errors		*Error Type*	*ITP*
Major	**Minor**		**SECTION**
0	0	Declaration, Definition and Reference Errors	
0	0	Control Flow Errors	
0	0	Computational Errors	
0	0	Task Interaction Errors	
0	0	Other Errors	
0	**0**	**Totals**	

Notes

_____

_____

_____