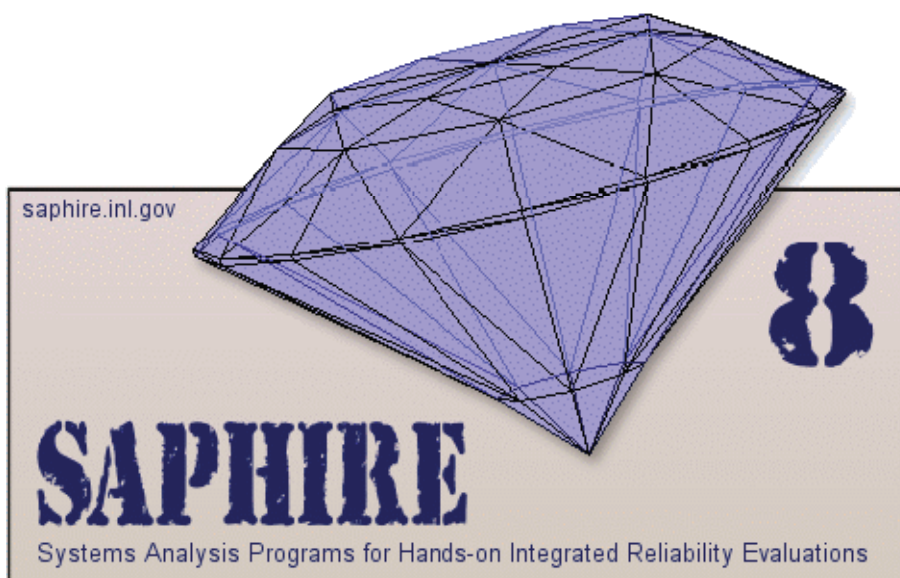


# Independent Verification and Validation of **SAPHIRE 8** Software Design and Interface Design

October 2009



The INL is a U.S. Department of Energy National Laboratory  
operated by Battelle Energy Alliance

# **Independent Verification and Validation of SAPHIRE 8 Software Design and Interface Design**

October 2009

**Idaho National Laboratory  
Idaho Falls, Idaho 83415**

**<http://www.inl.gov>**

**Prepared for the  
U.S. Nuclear Regulatory Commission  
Washington, DC 20555  
Project Number N6423**

## Table of Contents

1.0	Executive Summary .....	1
2.0	Background Information .....	2
3.0	Summary of Findings.....	3
3.1	NUREG/BR-0167 Findings .....	3
3.1.1	Section 2.3 Design .....	3
3.1.2	Section 2.6 Installation and Acceptance .....	3
3.1.3	Section 3.1 Concepts and Definitions .....	4
3.1.4	Section 3.2.2 Formal Life Cycle Reviews and Audits.....	4
3.1.4.1	Section 3.2.2.1 Software Requirements Review .....	4
3.1.4.2	Section 3.2.2.2 Preliminary Design Review .....	4
3.1.4.3	Section 3.2.2.3 Critical Design Review .....	5
3.1.4.4	Section 3.2.2.4 Qualification Test Readiness Review .....	5
3.1.4.5	Section 3.2.2.5 Software Configuration Audit.....	5
3.1.4.6	Section 3.2.2.6 Post Mortem Review.....	5
3.1.5	Section 3.2.3 Formal Peer Inspections.....	5
3.1.6	Section 3.2.4.1 Design-Driven and Requirements-Driven Testing .....	6
3.1.7	Section 4.3 Software Requirements Documentation .....	6
3.1.8	Section 4.4 Software Design Documentation .....	7
3.1.9	Section 6 Configuration Management .....	7
3.1.10	Section 6.2 Baselines .....	8
3.1.11	Appendix B Test Plan .....	9
3.2	Section 4 Design Specification Findings .....	9
3.3	Section 5 Test Specification Findings.....	11
3.4	Section 6 Test Methodologies Findings.....	11
4.0	IV&V Evaluation Checklist.....	13

## **1.0 Executive Summary**

The purpose of the Independent Verification and Validation (IV&V) role in the evaluation of the SAPHIRE software design and interface design is to assess the activities that results in the development, documentation, and review of a software design that meets the requirements defined in the software requirements documentation. The IV&V team began this endeavor after the software engineering and software development of SAPHIRE had already been in production. IV&V reviewed the requirements specified in the NRC Form 189s to verify these requirements were included in SAPHIRE's Software Verification and Validation Plan (SVVP) design specification.

The requirements for IV&V review were extracted primarily from the NUREG but also included an examination of best software engineering methods provided in the IEEE Standard for Software Verification and Validation. IV&V developed a checklist that mapped these requirements with these standards which was used in the evaluation. The evaluation criteria and the results of the assessment are identified in section 4 of this document.

Traceability of requirements is the greatest of these concerns. Requirements traceability is essential to all software development activities. Without a well documented Requirements Traceability Matrix (RTM), design components may be overlooked, and test cases missed.

For IV&V to properly evaluate the RTM to assess the mapping of the test cases to design components and to requirements as documented in SAPHIRE's Software Verification and Validation Plan, IV&V had to obtain requirements from the Statement of Work documents (Form 189s) and develop the RTM. This action could place IV&V's "independence" role into question. The intent of IV&V in developing the RTM is strictly for use in evaluation and not intended for use by the development team. However, the RTM will be included as documentary evidence in the IV&V report provided to the sponsor and the INL Project Manager.

Per the requirements and document outline provided in the SAPHIRE IV&V Plan, this report and all subsequent reports will be included as attachments and/or background evidence of the evaluation as well as the results of the assessment.

## **2.0 Background Information**

NUREG/BR-0167, Software Quality Assurance Program and Guidelines, requires the development of Software Design Documentation that specifies the overall structure of the software so that it can be translated into code. The Software Design Documentation includes a description of the major elements of the software as they relate to the requirements; a description of the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure; and an identification and detailed definition of the software units and data elements of the software architecture.

This report provides an evaluation of the Software Design Documentation. The Software Design Documentation is intended to provide the development, documentation, and review of the software design that meets the requirements defined in the Software Requirements Documentation to meet the contractual commitments prepared by the sponsor; the Nuclear Regulatory Commission.

Independent Verification and Validation (IV&V) evaluates and assesses the processes and products developed during each phase of the Software Development Life Cycle (SDLC). The SAPHIRE 8 development team is implementing a “spiral” rapid application approach to the product development. One of the roles that IV&V performs, regardless of the development methodology, is to analyze products developed throughout the development process. The intent is to provide a level of confidence to the sponsor that the quality of the software product and supporting documentation is built into the software, not tested in. Evaluating the supporting documentation for each product is one aspect of providing this level of confidence.

IV&V supports and is complementary to the Quality Assurance, Project Management, and product development activities. To achieve this support, IV&V must also evaluate the processes identified in the documentation to ensure that the development team is implementing the processes and methodology that ensures a high-level software product.

Due to the spiral approach implemented for the software development, it is expected that the Software Design Documentation will evolve as the SAPHIRE 8 product matures. Therefore, IV&V will evaluate each iteration of the Software Design Documentation.

To provide direction in the evaluation process, IV&V has developed a checklist to support the requirements for the SDLC. The Project Plan requirements used for the analysis of the Software Design Documentation is contained in a checklist that is included in the SAPHIRE 8 Software Independent Verification and Validation Plan (INL/EXT-09-15649, Revision ID: 0, Effective Date: April 1, 2009). The evaluation criteria for the Software Design Documentation have been extracted from the checklist contained in the “IV&V Plan” and included in section 4 of this report. A summary of the findings is provided in section 3.

### **3.0 Summary of Findings**

The design is included in section 4 Design Specification within the SAPHIRE Version 8 Software Verification and Validation Plan Volume I. A requirements traceability analysis was performed to trace design components to requirements, and requirements and design components to test cases. The results of the requirements traceability analysis is in attachment NRC Form 189 Requirements Table.xls.

There is no Software Design Documentation containing all design components that meet the requirements.

The software design does not completely meet the requirements defined in the SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 2 Software Requirements and section 3 Interface Requirements Specification. The software design needs to specify the overall structure of the software so that it can be translated into code. The consistency of detail between design components and software requirements varies.

The design components are not all documented or uniquely identified, but design components that are documented are specified in paragraphs and referenced in the Requirements Traceability Matrix as section numbers within the SAPHIRE Version 8 Software Verification and Validation Plan Volume I.

#### **3.1 NUREG/BR-0167 Findings**

##### **3.1.1 Section 2.3 Design**

Fail

The design process is the set of activities that results in the development, documentation, and review of a software design that meets the requirements defined in the software requirements documentation.

As the design evolves, events (e.g., additional insight into problem areas) may necessitate the modification of the requirements documentation. Manage changes to requirements documentation in accordance with well defined change control procedures.

##### **3.1.2 Section 2.6 Installation and Acceptance**

Fail

Acceptance activities include:

1. Execution of acceptance tests (which typically consist of some of the qualification test cases plus additional test cases).
2. Documentation of the acceptance of the software by the sponsor.

This stage of the life cycle usually concludes with the user accepting the software for operational use. The responsibility for the sustaining engineering and maintenance of the software may be assigned to an organization different from the sponsor and/or the developer of the software.

### **3.1.3 Section 3.1 Concepts and Definitions**

Fail

Examples of verification activities include:

1. Formal major life cycle reviews and audits (e.g., Preliminary Design Review).
2. Formal peer inspections (e.g., code inspections, documentation reviews).
3. Informal tests (e.g., unit and integration testing).

### **3.1.4 Section 3.2.2 Formal Life Cycle Reviews and Audits**

Fail

A formal review, with sponsor and developer management and technical personnel participating, is held at or near the end of a major activity of the life cycle. The objective of the formal reviews is to evaluate the deliverable products, the progress, and to a lesser degree, the processes of the most recent life-cycle phase.

The products associated with each formal review are:

1. The documents to be reviewed (e.g., the software requirements documentation for the Software Requirements Review).
2. The agenda for the review.
3. The hardcopy presentation materials.
4. The minutes that document the activities and results of the review.
5. The updated documents that were reviewed.

Allow sufficient time for sponsor review participants to review the documents prior to the review (2 to 3 weeks, say). Identify in the agenda the review participants and their specific responsibilities during the review. Assign a person to capture key discussion items and actions items, especially those related to specific assignments for updating the documentation that is the object of the review. Document in the review minutes all proposed revisions to the reviewed documents and all actual changes to the reviewed documents, and place the updated documents under configuration control after approval by the sponsor.

#### **3.1.4.1 Section 3.2.2.1 Software Requirements Review**

Fail

Conduct the Software Requirements Review at the end of requirements definition. The primary objective of this review is to assure that the sponsor and the developer understand and agree on the intent, completeness, verifiability (through testing or other means), consistency, and technical feasibility of the requirements. Secondary objectives are to review other documentation products available at this time, including, for example, the Software Project Plan and the overall verification and validation plan.

#### **3.1.4.2 Section 3.2.2.2 Preliminary Design Review**

Fail

Conduct the Preliminary Design Review when the preliminary design (software architecture) has been designed. The primary objective of this review is to assure that the preliminary design is

complete (meets all the requirements), verifiable (through testing or other means), consistent, and technically feasible.

#### **3.1.4.3      Section 3.2.2.3 Critical Design Review**

Fail

Conduct the Critical Design Review when the design is complete. Design completion criteria should be defined clearly in the Software Project Plan. Suggested design completion criteria are:

1. All software units have been identified and all interfaces between and among the units have defined.
2. All elements of the database have been defined down to the data item level.

The primary objective of this review is to assure that the design is complete (meets all the requirements and meets design completion criteria), verifiable (through testing or other means), consistent, and technically feasible.

#### **3.1.4.4      Section 3.2.2.4 Qualification Test Readiness Review**

N/A

Conduct the Qualification Test Readiness Review when integration testing and the qualification test procedures are complete. The primary objective of this review is to assure that the as-built software; the software documentation; and qualification test data, test tools, test configuration, and test team are ready for formal qualification testing.

#### **3.1.4.5      Section 3.2.2.5 Software Configuration Audit**

Fail

Conduct the Software Configuration Audit twice, first at the completion of qualification testing and second at the completion of acceptance testing. The primary objective of this audit is to ensure that the as-built software:

1. Meets its requirements as documented in the software requirements documentation.
2. Conforms to its technical documentation.
3. Does not contain any unauthorized changes.

#### **3.1.4.6      Section 3.2.2.6 Post Mortem Review**

N/A

Conduct the Post Mortem Review after the software has been accepted. The objective of this audit is to capture the lessons learned from the project for use by future projects.

### **3.1.5 Section 3.2.3 Formal Peer Inspections**

Fail

A formal peer inspection is a detailed examination of a product on a step-by-step or line-by-line basis. The purpose of conducting formal peer inspections is to find errors. The group that performs a peer inspection is composed of peers of the person who developed the product to be



inspected. Peer inspections are objective approaches that have been proved very effective in verifying that products meet requirements.

For Level 1 software, require the developer to

1. Subject each intermediate product and final product of development and maintenance (i.e., all documentation, all code) to an internal peer inspection.
2. Make available to the sponsor the written procedure and the product standards that govern peer inspections.
3. Make available, if requested by the sponsor, records that document the results of all peer inspections.

For Level 2 software, encourage the developer to work toward subjecting each intermediate product and final product of development and maintenance to an internal peer inspection.

### **3.1.6 Section 3.2.4.1 Design-Driven and Requirements-Driven Testing**

Fail

Design-driven or white-box testing is the process where the tester examines the internal workings of the code. Design-driven testing is accomplished by selecting input data and other parameters based on the internal logic paths to be checked. The goals of design-driven testing include ascertaining correctness of:

1. All paths through the code. For most software products, this can be feasibly done only at the unit test level.
2. Interfaces between units.
3. Size and timing of critical elements of code.

Requirements-driven or black-box testing is done by selecting input data and other parameters based on the software requirements and observing the outputs and reaction of the software. In addition to testing for satisfaction of requirements, some of the objectives of requirements-driven testing are to ascertain:

1. Computational correctness.
2. Proper handling of boundary conditions, including extreme inputs and conditions that cause extreme outputs.
3. State transitioning as expected.
4. Proper behavior under stress or high load.
5. Adequate error detection, handling, and recovery.

Sometimes the term "operational testing" is used. Operational testing is either the random or statistically controlled application of the software in its actual environment or in a simulated version of the operational environment. An example of such testing is the so-called beta test use of an applications software package by individuals typical of the intended user population. In the terminology used above, such operational testing and beta testing would be qualification testing and are requirements-driven.

### **3.1.7 Section 4.3 Software Requirements Documentation**

Fail

Software requirements documentation specifies the requirements that the software to be developed/maintained must meet. Include in this documentation the following, as applicable:

1. Functionality – the functions that the software is to perform.
2. Performance – the time-related requirements of software operation such as speed, response time, etc.
3. Design constraints imposed on implementation activities – any elements that will restrict design options (e.g., specifying the hardware platform or the programming language).
4. Attributes – characteristics of the software, its acceptance, or use (e.g., portability, acceptance criteria, access control, availability, maintainability, etc.).
5. External interfaces – interactions with people, hardware, and other software.

An item can be called a software requirement only if its achievement can be verified and validated. It is important that each software requirement be traceable throughout the stages of the software life cycle.

### **3.1.8 Section 4.4 Software Design Documentation**

Fail

In software design documentation, specify the overall structure of the software so that it can be translated into code. Include in this documentation:

1. A description of the major elements of the software as they relate to the requirements.
2. A description of the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure.
3. An identification and detailed definition of the software units and data elements of the software architecture.

### **3.1.9 Section 6 Configuration Management**

Pass

For a project to be successful, the developer and sponsor must establish and maintain integrity of the software and its documentation as they evolve throughout the life cycle. Because requirements, the design, the code, and the test environment can change significantly and often, it is essential that change be managed successfully. Briefly stated, configuration management is change management.

Fundamental to configuration management are the concepts of a baseline and change control. A baseline is a document or software that has been formally reviewed and agreed upon by the developer and sponsor, that thereafter serves as the basis for further development and that can be changed only through formal change control procedures. Change control is the process by which a change to a baseline is proposed, evaluated, approved or rejected, scheduled, and tracked.

There are four major functions of configuration management:

1. The identification and establishment of baselines.
2. Controlling both changes to baselines and the release of baselines.
3. Recording and reporting the status of baselines, change requests, and implemented changes.

4. Verifying, through auditing, the correctness and completeness of baselines prior to release.

For a software configuration management program to be successful, experience has shown that most of the following conditions exist:

1. The content and status of the software and documentation baselines are known at all times.
2. The developer follows a written configuration management policy that has the following characteristics:
  - (a) Responsibility for configuration management for each project is explicitly assigned.
  - (b) Configuration management is implemented on products throughout the product's life cycle.
  - (c) Configuration management is implemented for externally-deliverable products and for appropriate products used inside the organization.
  - (d) All projects have a repository for storing key software engineering elements and associated configuration management records.
  - (e) The software baselines and configuration management activities are audited on a regular basis.
3. A group that is responsible for coordinating and implementing configuration management for the project exists or is established.
4. Adequate resources and budget for performing configuration management activities are provided.
5. Members of the configuration management group are trained in the objectives, procedures, and methods for performing their assigned activities.
6. The configuration management activities are reviewed with the project manager on a regular basis.

### **3.1.10 Section 6.2 Baselines**

Pass

Establish controlled and stable baselines for planning, managing, and building the system. Explicitly identify as project baselines software products (e.g., source code, object code, test cases) and software process specifications (e.g., standards and procedures) that are needed to establish and maintain stability of the software activities.

Establish a naming or labeling system that:

1. Uniquely identifies all project entities (e.g., documents, software elements, test cases).
2. Identifies changes by revision or version.
3. Uniquely identifies each configuration/version of revised software for use.

Establish the following baselines:

1. The project management baseline consisting of the Software Project Plan, documented standards and procedures, and up-to-date budgets and schedules.

2. The requirements baseline consisting of the software requirements documentation plus approved changes.
3. The product baseline consisting of software and documentation resulting from the qualification testing activity.
4. The operational baseline consisting of software and documentation resulting from the installation and acceptance activity that is placed into operation.

The developmental configuration is the developer's software and associated technical documentation that defines the evolving software products during development. It contains the software design and implementation products (software design documentation, code, test cases, and related information). Require the developer to apply internal configuration control procedures to the developmental configuration as it evolves.

### **3.1.11 Appendix B Test Plan**

Fail

A document prescribing the approach to be taken for intended testing activities. The plan typically identifies the items to be tested, the requirements being tested, the testing to be performed, test schedules, personnel requirements, reporting requirements, evaluation criteria, any risks requiring contingency planning.

## **3.2 Section 4 Design Specification Findings**

The following list of requirements from the SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 2 Software Requirements and section 3 Interface Requirements Specification could not be traced to specific design specifications within the SAPHIRE Version 8 Software Verification and Validation Plan Volume I Section 4 Design Specification (Refer to attachment NRC Form 189 Requirements Table.xls Requirements tab for requirements):

- R-01
- R-02
- R-05
- R-07
- R-08
- R-15
- R-17
- R-20
- R-24
- R-29
- R-31
- R-32
- R-33
- R-34
- R-35
- R-36
- R-38
- R-48

- R-50
- R-51
- R-52
- R-53
- R-54
- R-55
- R-60
- R-63
- R-66
- R-67
- R-73
- R-74

The following list of sections from the SAPHIRE Version 8 Software Verification and Validation Plan Volume I Section 4 Design Specification could not be traced to specific requirements within the SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 2 Software Requirements and section 3 Interface Requirements Specification:

- DC-09 - Section 4.1.8 External Events Design
- DC-10 - Section 4.1.8.1 General Model Type Information
- DC-13 - Section 4.1.8.4 Analysis Capabilities Specific to EE Models
- DC-14 - Section 4.1.8.5 Showing Model Types for User Interaction
- DC-15 - Section 4.1.9 Electronic Help Design
- DC-15 - Section 4.1.9.1 Nominal HTML Based Help Structure
- DC-15 - Section 4.1.9.2 General Structure of the Help Page
- DC-15 - Section 4.1.9.3 Calling the Help File
- DC-21 - Section 4.4.2 Event Object Design
- DC-22 - Section 4.4.3 Fault Tree Object Design
- DC-23 - Section 4.4.4 Event Tree Object Design
- DC-43 - Section 4.5.9 General Analysis Design
- DC-44 - Section 4.5.10 User Defined Model Type Analysis Design
- DC-46 - Section 4.6 Embedded Macro Design
- DC-47 - Section 4.7 Phase Mission Design
- DC-48 - Section 4.7.1 General Phase Information
- DC-49 - Section 4.7.2 Basic Event Phase Modifications
- DC-50 - Section 4.7.3 Fault Tree Phase or Applicability Modifications
- DC-51 - Section 4.7.4 Event Tree Phase or Applicability Modifications
- DC-52 - Section 4.7.5 Analysis Capabilities Phase Modifications

The following list of sections from the SAPHIRE Version 8 Software Verification and Validation Plan Volume I Section 4 Design Specification are incomplete:

- Section 4.1.2 Project controls, contains no information.
- Section 4.1.3 Project Integration and Modification specifies, *“This functionality is currently under development”*.

- Section 4.1.5 Project-Wide Search Design specifies, “*This functionality is currently in the concept stage of design*”.
- Section 4.1.8 External Events Design, contains no information.
- Section 4.4.5 End State Object Design specifies, “*End state objects are currently under development*”.
- Section 4.5.6 Cut Set Analysis Design, Editing Cut Sets bullet specifies, “*This feature is currently in the concept stage of design*”.
- Section 4.5.9 General Analysis Design specifies, “*This analysis is currently in the development stage of design*”.

### 3.3 Section 5 Test Specification Findings

SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 5.1 Features to be Tested last bullet states “Verification of the new user interface. (Manual tests at this point in time). There does not appear to be any tests specific that test the new user interface.

SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 5.1 Features to be Tested contain Table 5-1 Tests where specific PRA features are verified. This table is incomplete in listing SAPHIRE Option, and Test Models. Specific findings are listed below.

- First row beginning with Test-01 and ending with Test-49. The Test Models column contains “All”. This column should list all Test Models that are used.
- Test-13 Project Uncertainty, Test-41 Cut Set Verification. The SAPHIRE Option and Test Models columns are blank.
- Row beginning with Test-05 and ending with Test-12. The Test Models column contains “All Models”. This column should list all Test Models that are used.
- Test-41 Cut Set Verification. The Test Models column contains “All”. This column should list all Test Models that are used.
- Test-51 Change Set Processing-Class. The “?” in line “Class change - ? –MOV-1 events, probability 0.5” needs clarification.
- Rows beginning with Test-03 and ending with Test-04. The Test Models column contains “All Models”. This column should list all Test Models that are used.
- Test-58 Base Case Updates specifies, “*This test suite was never run*”. The Test Models column contains “All SPAR 2Q, 3i models”. This column should list all Test Models that are used.
- Test-60 Change Sets. The PRA Area column is blank. The Test Models column contains “TBD”.
- Test-61 Uncertainty analysis. The PRA Area column is blank.
- Test-64 specifies “*DOES NOT EXIST YET*”.
- Test-65 Event Transformations. The Test Models column contains “TBD”.

### 3.4 Section 6 Test Methodologies Findings

The following list of requirements from the SAPHIRE Version 8 Software Verification and Validation Plan Volume I section 2 Software Requirements and section 3 Interface Requirements Specification could not be traced to specific test cases within the SAPHIRE Version 8 Software Verification and Validation Plan Volume II Section 6 Test Methodologies (Refer to attachment NRC Form 189 Requirements Table.xls Requirements tab for requirements):

- R-01
- R-02
- R-07
- R-15
- R-24
- R-29
- R-30
- R-31
- R-37
- R-63
- R-65
- R-66
- R-67
- R-68
- R-70
- R-73
- R-74

The following list of design specifications from the SAPHIRE 8 Software Verification and Validation Plan Volume I Section 4 Design Specification could not be traced to specific test cases within the SAPHIRE Version 8 Software Verification and Validation Plan Volume II Section 6 Test Methodologies:

- DC-01 - Section 4.1 Graphical User Interface (GUI) Design
- DC-02 - Section 4.1.1 Access to Top Level Objects Design
- DC-07 - Section 4.1.6 User Selectable Constants Design

The following list of sections from the SAPHIRE Version 8 Software Verification and Validation Plan Volume II Section 6 Test Methodologies are incomplete:

- Section 6.2.1.62 Test-62, N of M Gates is not listed in Table 5-1 Tests where specific PRA features are verified.
- Section 6.2.1.63 Test-63, Sequence Stress Testing is not listed in Table 5-1 Tests where specific PRA features are verified.
- Section 6.2.1.65 Test-65, Event Transformations specifies, *“This test is currently in the concept design phase”*.

The SAPHIRE Version 8 Software Verification and Validation Plan Volume II Section 6 Test Methodologies contain Table 6.2.1-1 Features tested by the automated test suite. The columns containing Test-60 and Test-65 list “TBD” as the Plant Model Used. Both columns are blank and do not reference any of the items in the first column that list the Features Tested. The rows “Event Tree linking” and “Calculation Types” do not appear to be tested by the automated test suite according to the table.



## 4.0 IV&V Evaluation Checklist

SOFTWARE DESIGN and INTERFACE DESIGN		
Criteria Priority: 1	Does the Software Design Specification (SDS) present the structure of the software such that it can be translated into code? NUREG/BR-0167 Section 4.4	
Pass		Comments
Fail	X	Major elements of the software as they relate to the requirements are missing. Sections describing the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure are not present.
N/A		
		Refer to section 3.0 Summary of Findings.
Criteria Priority: 1	Does the SDS provide a description of the major elements/components of the software as related to the requirements in the SRS? NUREG/BR-0167 Section 4.4	
Pass		Comments
Fail	X	The software design specification contains design specification for requirements that are not uniquely identified in the software requirements documentation and is missing software design specification for requirements that are in the software requirements documentation.
N/A		
		Refer to section 3.0 Summary of Findings.
Criteria Priority: 1	Does the SDS provide a technical description in terms of the theoretical basis? NUREG/BR-0167 Section 4.4	
Pass		Comments
Fail	X	Sections describing the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure are not present.
N/A		
		Refer to section 3.0 Summary of Findings.
Criteria Priority: 1	Does the SDS provide a technical description in terms of the mathematical model? NUREG/BR-0167 Section 4.4	
Pass		Comments
Fail	X	Sections describing the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure are not present.
N/A		
		Refer to section 3.0 Summary of Findings.
Criteria Priority: 1	Does the SDS provide a technical description of the data flow(s) and data structure(s)? NUREG/BR-0167 Section 4.4	
Pass		Comments
Fail	X	Sections describing the theoretical basis, physical model, mathematical model, control flow, data flow, control logic, and data structure are not present.
N/A		
		Refer to section 3.0 Summary of Findings.
Criteria Priority: 1	Does the SDS provide the defined range of input values? NUREG/BR-0167 Section 3.2.4.1 (boundary conditions)	
Pass		Comments
Fail	X	Various discussions of inputs are found throughout section 4 Design Specification. A section providing the defined range of input values is not present.
N/A		
		Refer to section 3.0 Summary of Findings.



<b>Criteria Priority: 1</b>	<b>Does the SDS provide the defined range of output values? NUREG/BR-0167 Section 3.2.4.1 (boundary conditions)</b>	
Pass		Comments
Fail	X	Various discussions of outputs are found throughout section 4 Design Specification. A section providing the defined range of output values is not present.
N/A		
		Refer to section 3.0 Summary of Findings.
<b>Criteria Priority: 1</b>	<b>Has the “Test Plan” and “Test Suite” for validating the software (by the development team) been addressed? NUREG/BR-0167 Appendix B</b>	
Pass		Comments
Fail	X	Section 4.6 Embedded Macro Design references the previously used Microsoft Visual Test suite, but no reference to the hardware platform used for testing. Section 5 Test Specification lists the features to be tested and features not tested. No reference is made to address the actual Test Plan and Test Suite for validating the software in section 4 Design Specification.
N/A		
		Refer to section 3.0 Summary of Findings.
<b>Criteria Priority: 1</b>	<b>Has the RTM been updated to map the design components back to the defined requirements and are the design components/requirements mapped to test cases? NUREG/BR-0167 Section 4.3</b>	
Pass		Comments
Fail	X	The Requirements Traceability Matrix is incomplete. Section numbers specified in the RTM do not match section numbers within section 4 Design Specification. There are no individual test cases mapped to requirements and design components.
N/A		
		Refer to section 3.0 Summary of Findings.
<b>Criteria Priority: 1</b>	<b>Has the acceptance criteria for specifying how to determine the validity of the software provided, given the results of the test cases? NUREG/BR-0167 Section 2.6</b>	
Pass		Comments
Fail	X	Section 6.3 Test Data and section 7.3 Sources of Data refer to acceptance criteria in general for all tests. Specific acceptance criteria given the results of each test case are not found.
N/A		
		Refer to section 3.0 Summary of Findings.
<b>Criteria Priority: 1</b>	<b>Are the test case identifiers unique/unambiguous? NUREG/BR-0167 Section 6.2, 2.6.2</b>	
Pass	X	Comments
Fail		Test case identifiers are Test-01 through Test-66.
N/A		
<b>Criteria Priority: 3</b>	<b>Has a data dictionary been developed? Software Engineering Practices</b>	
Pass		Comments
Fail	X	Develop the Data Dictionary defining items computed in the code. What they are and what they do.
N/A		
<b>Criteria Priority: 2</b>	<b>If the SRS is found to require an update, has the SRS been updated, information represented correctly, completely, and accurately in the SRS? NUREG/BR-0167 Section 4.3, Section 6</b>	
Pass		Comments
Fail	X	Refer to document Independent Verification and Validation Of SAPHIRE 8 Software Requirements Project Number: N6423 U.S. Nuclear Regulatory Commission, Document ID: INL/EXT-09-16789.
N/A		

Criteria Priority: 1	Have all documents, including revised documents from the Requirements phase, been placed under Configuration Control and were Configuration Control procedures been performed completely and accurately? NUREG/BR-0167 Section 6	
Pass	X	Comments
Fail		The requirements are included in section 2 Software Requirements and section 3 Interface Requirements Specification, the design is in section 4 Design Specification and section 5 Test Specification is within the SAPHIRE Version 8 Software Verification and Validation Plan Volume I. Section 6 Test Methodologies is within the SAPHIRE Version 8 Software Verification and Validation Plan Volume II.
N/A		
Criteria Priority: 1	Have Peer Reviews, Software Requirements Reviews, Preliminary Design Reviews, Critical Design Reviews and Qualification Readiness Reviews been performed, with recorded results (usually via checklist or pre-approved form), and placed under configuration control? NOTE: IV&V activities require attendance at all major life-cycle reviews and audits. NUREG/BR-0167 Section 3.1 and 3.2.2, 3.2.3	
Pass		Comments
Fail	X	Refer to section 3.0 Summary of Findings.
N/A		