INL/EXT-09-17014

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 6 Quality Assurance

C. L. Smith
R. Nims
K. J. Kvarfordt

March 2011

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

INL/EXT-09-17014
NUREG/CR-7039

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 6 Quality Assurance

C. L. Smith
R. Nims
K. J. Kvarfordt

March 2011

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555
Job Code N6423

AVAILABILITY NOTICE

Availability of Reference Materials Cited in NRC Publications

Most documents cited in NRC publications will be available from one of the following sources:

1. The NRC Public Document Room, Rockville Pike, Rockville, MD 20852 (pdr@nrc.gov)

2. The Superintendent of Documents, U. S. Government Printing Office (GPO), Mail Stop SSOP, Washington, DC 20402-9328

3. The National Technical Information Service, Springfield, VA 22161

Although the listing that follows represents the majority of documents cited in NRC publications, it is not intended to be exhaustive.

Referenced documents available for inspection and copying for a fee from the NRC Public Document Room include NRC correspondence and internal NRC memoranda; NRC bulletins, circulars, information notices, inspection and investigative notices; licensee event reports; vendor reports and correspondence; Commission papers; and applicant and licensee documents and correspondence.

The following documents in the NUREG series are available for purchase from the GPO Sales Program: formal NRC staff and contractor reports, NRC-sponsored conference proceedings, international agreement reports, grant publications, and NRC booklets and brochures. Also available are regulatory guides, NRC regulations in the *Code of Federal Regulations*, *and Nuclear Regulatory Commission Issuances*.

Documents available from the National Technical Information Service include NUREG-series reports and technical reports prepared by other Federal agencies and reports prepared by the Atomic Energy Commission, forerunner agency to the Nuclear Regulatory Commission.

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions. *Federal Register* notices, Federal and State legislation, and congressional reports can usually be obtained from these libraries.

Documents such as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings are available for purchase from the organization sponsoring the publication cited.

Single copies of NRC draft reports are available free, to the extent of supply, upon written request to the Office of Administration, Distribution and Mail Services Section U. S. Nuclear Regulatory Commission, Washington, DC 20555-0001.

The public maintains copies of industry codes and standards used in a substantive manner in the NRC regulatory process at the NRC Library, Two White Flint North, 11545 Rockville Pike, Rockville, MD, 20852, for use. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from the American National Standards Institute, 1430 Broadway, New York, NY 10018.

DISCLAIMER NOTICE

PREVIOUS REPORTS

S. T. Wood, C. L. Smith, K. J. Kvarfordt, S. T. Beck, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 1 Summary Manual*, NUREG/CR-6952, August 2008.

C. L. Smith, S. T. Wood, W. J. Galyean, J. A. Schroeder, S. T. Beck, M. B. Sattison, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 2 Technical Reference*, NUREG/CR-6952, August 2008.

K. J. Kvarfordt, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 3 Code Reference Manual*, NUREG/CR-6952, August 2008.

S. T. Beck, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 4 Tutorial*, NUREG/CR-6952, August 2008.

C. L. Smith, J. Schroeder, S. T. Beck, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 5 GEM Manual*, NUREG/CR-6952, August 2008.

C. L. Smith, R. Nims, K. J. Kvarfordt, C. Wharton, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 6 Quality Assurance Manual*, NUREG/CR-6952, August 2008.

K. J. Kvarfordt, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 7 Data Loading*, NUREG/CR-6952, August 2008.

Smith, C. L., et al., *Testing, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0*, NUREG/CR-6688, October 2000.

K. D. Russell, et al. *Systems Analysis Programs for Hands-on Reliability Evaluations (SAPHIRE) Version 6.0 - System Overview Manual*, NUREG/CR-6532, May 1999.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 5.0, Volume 2 - Reference Manual*, NUREG/CR-6116, EGG-2716, July 1994.

K. D. Russell et al., *Verification and Validation (V&V), Volume 9 – Reference Manual*, NUREG/CR-6116, EGG-2716, July 1994.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 4.0, Volume 1 - Reference Manual*, NUREG/CR-5813, EGG-2664, January 1992.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 2.5 Reference Manual*, NUREG/CR-5300, EGG-2613, March 1991.

K. D. Russell, M. B. Sattison, D. M. Rasmuson, *Integrated Reliability and Risk Analysis System (IRRAS) - Version 2.0 User's Guide*, NUREG/CR-5111, EGG-2535, manuscript completed March 1989, published June 1990.

K. D. Russell, D. M. Snider, M. B. Sattison, H. D. Stewart, S.D. Matthews, K. L. Wagner, *Integrated Reliability and Risk Analysis System (IRRAS) User's Guide - Version 1.0 (DRAFT)*, NUREG/CR-4844, EGG-2495, June 1987.

ABSTRACT

The Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8 is a software application developed for performing a complete probabilistic risk assessment using a personal computer running the Microsoft Windows™ operating system. SAPHIRE 8 is funded by the U.S. Nuclear Regulatory Commission (NRC). The role of the INL in this project is that of software developer and tester. This development takes place using formal software development procedures and is subject to quality assurance (QA) processes. The purpose of this document is to describe how the SAPHIRE software QA is performed for Version 8, what constitutes its parts, and limitations of those processes. In addition, this document describes the Independent Verification and Validation that was conducted for Version 8 as part of an overall QA process.

FOREWORD

The U.S. Nuclear Regulatory Commission (NRC) has developed the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) software that is used to perform probabilistic risk assessments (PRAs) on a personal computer. SAPHIRE enables users to supply basic event data, create and solve fault and event trees, perform uncertainty analyses, and generate reports. In that way, analysts can perform PRAs for any complex system, facility, or process.

For nuclear power plant PRAs, SAPHIRE can be used to model a plant's response to initiating events, quantify core damage frequencies, and identify important contributors to core damage (Level 1 PRA). The program also can be used to evaluate containment failure and release models for severe accident conditions given that core damage has occurred (Level 2 PRA). In so doing, the analyst could build the PRA model assuming that the reactor is initially at full power, low power, or shutdown. In addition, SAPHIRE can be used to analyze both internal and external events and, in a limited manner, to quantify the frequency of release consequences (Level 3 PRA). Because this software is a very detailed technical tool, users should be familiar with PRA concepts and methods used to perform such analyses.

SAPHIRE has evolved with advances in computer technology and users' needs. Starting with Version 5, SAPHIRE operated in the Microsoft Windows™ environment. Versions 6 and 7 included features and capabilities for developing and using larger, more complex models. SAPHIRE Version 8 includes significant new features and capabilities to meet user needs for NRC risk-informed programs. In general, these include:

Improved user interfaces supporting NRC's Significance Determination Process, event and condition assessments, and more detailed types of PRA analyses.

Development and use of NRC's Standardized Plant Analysis Risk models.

New and improved solving algorithms.

Support features for user-friendliness.

This NUREG-series report comprises seven volumes as outlined below and incorporates new features and capabilities of Version 8.

Volume 1, "Overview and Summary"

Volume 1 provides an overview of the functions and features available in SAPHIRE Version 8 and presents general instructions for using the software.

Volume 2, "Technical Reference"

Volume 2 summarizes the fundamental mathematical concepts of sets and logic, fault trees, and probability. It then describes the algorithms used to construct a fault tree and to obtain the minimal cut sets. This report presents the formulas used to obtain the probability of the top event from the minimal cut sets and the formulas for probabilities that apply for various assumptions concerning reparability and mission time. In addition, it defines the measures of basic event importance that SAPHIRE can calculate. This volume also gives an overview of uncertainty analysis using simple Monte Carlo sampling or Latin Hypercube sampling and states

the algorithms used by this program to generate random basic event probabilities from various distributions. Finally, this report discusses enhanced and new capabilities such as post-processing rules, integrated model solving using model types, and workspace analysis routines.

Volume 3, "Users' Guide"

Volume 3 provides a brief discussion of the purpose and history of the software as well as general information such as installation instructions, starting and stopping the program, and some pointers on how to get around inside the program. Next, it discusses database concepts and structure. The following nine sections (one for each of the menu options on the SAPHIRE main menu) furnish the purpose and general capabilities for each option. Finally, Volume 3 provides the capabilities and limitations of the software.

Volume 4, "Tutorial"

Volume 4 provides a series of lessons that guide the user through basic steps common to most analyses performed with SAPHIRE.

Volume 5, "Workspaces"

Volume 5 describes the functionality and process behind SAPHIRE Version 8 workspaces. Workspaces provide an area in which a PRA model can be analyzed to obtain risk insights for a given initiating event or condition. Workspaces replace the "Graphical Evaluation Module" in earlier SAPHIRE versions.

Volume 6, "Quality Assurance"

Volume 6 is designed to describe how the SAPHIRE software quality assurance (QA) is performed for Version 8, what constitutes its parts, and the limitations of those processes. In addition, this report describes the Independent Verification and Validation that was conducted for Version 8 as part of an overall QA process.

Volume 7, "Data Loading"

Volume 7 is designed to guide the user through the basic procedures necessary to enter PRA data into the SAPHIRE program using SAPHIRE's MAR-D ASCII-text (or "flat file") data formats. In addition, this manual covers loading data through the new Accident Sequence Matrix and discusses the Project Integrate interfaces with SAPHIRE.

Christiana H. Lui, Director
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission

CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

Product quality is a key component of SAPHIRE Version 8. The SAPHIRE QA processes documented in the report provides the basis for setting quality objectives, progress, and the necessary framework for quality improvements. A majority of the changes within the SAPHIRE software occur because the end user has identified characteristics that provide "new potential", thus resulting in SAPHIRE evolving as each new feature is discovered and implemented. Therefore, the majority of software maintenance comes about not because of deficiencies in the code, but because it was modified to embrace improved methods for risk and reliability assessment.

Quality assurance for Version 8 has increased in a number of ways over the process described in NUREG/CR – 6688, "Testing, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0," September 2000. Early versions of SAPHIRE had utilized software guidance in some ways more rigorously than that used for Versions 6.0 and 7.0. However, this resulted in labor and time intensive testing and documentation practices. With the expansion of computer capabilities, automated testing for SAPHIRE significantly increased the number of tests which could be performed. The testing, verification, and validation process for Versions 6.0 and 7.0, therefore, emphasized automated testing, with decreased emphasis on maintaining formal documentation. Version 8 quality assurance effectively not only increases the test suite for its new features and capabilities, but also increases the formal documentation to ensure a quality product. Version 8 has also benefited from an independent verification and validation and extensive beta testing.

Version 8 follows guidance in NUREG/BR-0167, "Software Quality Assurance Program and Guidelines," February 1993 and IEEE Std 1010™-2004, "IEEE Standard for Software Verification and Validation." Per the guidance in NUREG/BR-0167, SAPHIRE is classified as "Level 1." Level 1 software corresponds to technical application software used in a safety decision. The IEEE Standard utilizes software integrity levels in its requirements, and SAPHIRE was assigned an integrity level of "1." This is the lowest level on a scale corresponding to the likelihood of occurrence of an operating state that contributes an error and an error consequence. The NRC periodically conducts a SAPHIRE audit against NUREG/BR-0167.

Documentation INL generated and maintains for Version 8 over earlier versions include:

- Software Verification and Validation Plan, including an associated requirements traceability matrix
- Design documents
- Software Project Plan
- System Test Plan
- Acceptance Test Plan
- Quality Assurance Plan

- Configuration Management Plan

In order to ensure the quality of the SAPHIRE software, the Idaho National Laboratory (INL) uses a variety of software development methods, including:

- Controlling software versions for both the formally released SAPHIRE versions, as well as for source code.
- Following a standard approach to bug fixes, implementing new features, and a release protocol.
- Developing design documents to control implementation of features.
- Using a cyclical design process to prototype changes.
- Performing acceptance tests that the software must pass prior to official release.
- Code walkthroughs and peer reviews.

The source code version control library requires that individual programmers “check-out” all files that they intend to modify. Prior to “check-in”, programmers must explain any changes made. A record is kept of all changes, both as explained by the developer, and as individual copies of each version of a file. At any time, the developer can retrieve past versions intact, if necessary. Since the SAPHIRE software program is continually modified, the version control procedure ensures a methodical approach to tracking and releasing these changes.

As new features and bug fixes are designed and implemented, the INL developers follow a standard approach to integrating these items into SAPHIRE. For bug fixes, the developers take notes from the user describing the general context of the bug, as well as step-by-step actions to reproduce the bugs. This bug information includes acquiring a copy of the user’s database, when necessary. Then, the bug is classified and prioritized according to severity. A bug is considered “minor” if it inconveniences the user, but a workaround exists to produce a correct answer. A bug is “major” if it prevents the user from obtaining the correct answer. Software enhancements follow much the same approach as bug fixes. Enhancements are prioritized and implemented, with intermediate testing by the developer and often by the requestor. Once the process and results appear acceptable, the feature is added to the next official release.

The level of effort for the software design process corresponds to the size and complexity of the proposed change. Developers use a cyclical prototyping design methodology as a means to clarify and refine the change. The prototyping process involves the requestor throughout development. The developers will interact with the requestor(s) both initially and throughout the design and development process to ensure the change accomplishes the expected goal.

Prior to any official SAPHIRE release of version 8, the software is run through a series of automated tests. The tests simulate user input to the computer through a test script, and results are captured and compared to expected results. This ensures that given a static input PRA file, the risk or reliability results from SAPHIRE will be consistent from one release to the next.

These acceptance tests were developed by first identifying the critical tasks performed in a PRA. Then these tasks were mapped to the SAPHIRE functions that perform these tasks. The critical functions were determined to include the following:

1. Fault tree analysis

2. Event tree and sequence analysis

3. End state analysis

4. Importance measures analysis

5. Uncertainty analysis

6. Change sets

7. Data utility functions

8. Workspace module functionality

Identification of tests is also assisted by learning from experience in using the software and finding bugs as well as reviewing the requirements specifications.

A change is not considered complete until the results have been tested and found reasonable. Developers and key users will test to see that the change works as expected and is free of defects. Prior to official release of a version, SAPHIRE's automated test suite must complete successfully. The success of the suite is a good indicator that the new change does not adversely affect other areas of the code.

ACKNOWLEDGEMENTS

We would like to specifically acknowledge Mr. Dan O'Neal of the U.S. Nuclear Regulatory Commission for his contribution to the development this report.

ACRONYMS

CDF	core damage frequency
DOE	Department of Energy
EF	error factor
FEP	Fault Tree, Event Tree, and Piping and Instrumentation Diagram
GEM	Graphical Evaluation Module
HRA	human reliability analysis
INL	Idaho National Laboratory
IRRAS	Integrated Reliability and Risk Analysis System
IV&V	Independent Verification and Validation
LHS	Latin Hypercube Sampling
MAR-D	Models and Results Database
NRC	Nuclear Regulatory Commission
PC	personal computer
PRA	probabilistic risk analysis
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SARA	System Analysis and Risk Assessment
SPAR	Standardized Plant Analysis Risk
V&V	Verification and Validation

Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 6 Quality Assurance

1. SAPHIRE QUALITY ASSURANCE PROCESSES

1.1 Background

The U.S. Nuclear Regulatory Commission (NRC) has developed a powerful personal computer (PC) software application for performing probabilistic risk assessments (PRAs), called Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8.

Using SAPHIRE 8 on a PC, an analyst can perform a PRA for any complex system, facility, or process. Regarding nuclear power plants, SAPHIRE can be used to model a plant's response to initiating events, quantify associated core damage frequencies, and identify important contributors to core damage (Level 1 PRA). It can also be used to evaluate containment failure and release models for severe accident conditions, given that core damage has occurred (Level 2 PRA). It can be used for a PRA assuming that the reactor is at full power, at low power, or at shutdown conditions. Furthermore, it can be used to analyze both internal and external initiating events, and it has special features for transforming models built for internal event analysis to models for external event analysis. It can also be used in a limited manner to quantify risk for release consequences to both the public and the environment (Level 3 PRA). For all of these models, SAPHIRE can evaluate the uncertainty inherent in the probabilistic models.

SAPHIRE development and maintenance has been undertaken by the Idaho National Laboratory (INL). The INL began development of a PRA software application on a PC in the mid 1980s when the enormous potential of PC applications started being recognized. The initial version, Integrated Risk and Reliability Analysis System (IRRAS), was released by the Idaho National Engineering Laboratory (now Idaho National Laboratory) in February 1987. IRRAS was an immediate success, because it clearly demonstrated the feasibility of performing reliability and risk assessments on a PC and because of its tremendous need (Russell 1987). Development of IRRAS continued over the following years. However, limitations to the state of the-art during those initial stages led to the development of several independent modules to complement IRRAS capabilities (Russell 1990; 1991; 1992; 1994). These modules were known as Models and Results Database (MAR-D), System Analysis and Risk Assessment (SARA), and Fault Tree, Event Tree, and Piping and Instrumentation Diagram (FEP).

IRRAS was developed primarily for performing a Level 1 PRA. It contained functions for creating event trees and fault trees, defining accident sequences and basic event failure data, solving system fault trees and accident sequence event trees, quantifying cut sets, performing sensitivity and uncertainty analyses, documenting the results, and generating reports.

MAR-D provided the means for loading and unloading PRA data from the IRRAS relational database. MAR-D used a simple ASCII data format. This format allowed interchange of data

between PRAs performed with different types of software; data of PRAs performed by different codes could be converted into the data format appropriate for IRRAS, and vice-versa.

SARA provided the capability to access PRA data and results (descriptive facility information, failure data, event trees, fault trees, plant system model diagrams, and dominant accident sequences) stored in MAR-D. With SARA, a user could review and compare results of existing PRAs. It also provided the capability for performing limited sensitivity analyses. SARA was intended to provide easier access to PRA results to users that did not have the level of sophistication required to use IRRAS.

FEP provided common access to the suite of graphical editors. The fault tree and event tree editors were accessible through FEP as well as through IRRAS, whereas the piping and instrumentation diagram (P&ID) editor was only accessible through FEP. With these editors an analyst could construct from scratch as well as modify fault tree, event tree, and plant drawing graphical figures needed in a PRA.

Previous versions of SAPHIRE consisted of the suite of these modules. Taking advantage of the Windows 95 (or Windows NT) environment, all of these modules were integrated into SAPHIRE Version 6; more features were added; and the user interface was simplified. With the release of SAPHIRE versions 5 and 6, INL included a separate module called the Graphical Evaluation Module (GEM). GEM provides a highly specialized user interface with SAPHIRE, automating SAPHIRE process steps for evaluating operational events at commercial nuclear power plants. GEM has been superseded in SAPHIRE Version 8 by way using customized Workspaces that provide specific kinds of analyses.

The development of the new SAPHIRE Version 8 includes new features and capabilities. These features and capabilities are related to working with larger, more complex models and improving the user-friendliness of SAPHIRE's interfaces while retaining key functionality of Version 7.

Version 8 is being developed to support NRC PRA models and to run them as an integrated model (e.g., Level 1 with external events). The graphical user interface has also improved from SAPHIRE 7 to support NRC programs such as the Significance Determination Process (SDP) and the Accident Sequence Precursor (ASP). A tailored interface for the SDP and the ASP programs has been developed. The interfaces for the SDP, ASP, and general analysis introduce the concept of a "workspace" in which the analyst may run and save different analyses. The use of workspaces enables the user to separate the model construction from the model analysis, thereby improving the quality of analysis being performed when using SAPHIRE.

The SAPHIRE Quality Assurance (QA) Manual provides the details to identify the methodology used to provide a planned and systematic approach required to guarantee the quality of the SAPHIRE software. To ensure the required quality is satisfied, the SAPHIRE development team applies the methodology needed to verify the design quality and to validate the software quality into the SAPHIRE software product. In addition, this document provides an overview into the general SAPHIRE QA process. Specifically, the report first outlines and describes the key part of the process. Second, the report discusses processes which address "building-in" quality assurance, and the formal testing program that is used to ensure software quality during the development cycle. Lastly, it concludes the report by reviewing the topics addressed.

In order to provide context to the complexity of a modern analysis code such as SAPHIRE (and its associated implications on testing), a list of salient features found in the software is provided in Appendix A. The combination of breadth and depth in these features shows the potential complexity that may be found in software as extensive as SAPHIRE.

Appendix B provides a template for a QA Checklist that is used to perform periodic inspections to monitor the SAPHIRE product quality. The checklist provides the identification for each inspection topic, an indication if the inspection, passed, failed, or was not applicable, as well as a column that may be used to insert specific comments regarding the inspection topic. Options for methods used to conduct the evaluation are random sampling, interviews, and observations. Assessment techniques can be modified to use more than one approach or a different approach than suggested in the checklist. The decision to use one or more techniques is conducted at the option of the evaluator.

In order to ensure quality of SAPHIRE, the important SAPHIRE features must be identified. Once these features are known, tests can be generated that would evaluate each feature. The results of these tests are described in Appendix C.

1.2 Summary of the SAPHIRE Quality Assurance

The SAPHIRE QA process encompasses several activities the INL uses to ensure quality throughout the development cycle. These activities are illustrated in Figure 1 and are described in this report.

Figure 1. SAPHIRE quality assurance process

As part of the overall QA process, the SAPHIRE TV&V process and results were previously documented in NUREG/CR-6688, *Testing, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0* (Smith et al, 2000) and NUREG/CR-6952, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 6 Quality Assurance Manual* (Smith et al, 2008). Within these documents, the existing TV&V was shown to depart from older V&V efforts (e.g., for versions 4 and 5) by focusing on the development and execution of a set of automated test scripts. For SAPHIRE 6.0 and 7.0, the TV&V process was expanded and automated so that the validity of the core functionality of SAPHIRE can be verified on an ongoing basis with each incremental release. For SAPHIRE 8, quality assurance activities not only included a significantly augmented test suite, but additional activities for compliance with applicable software guidance. These activities correspond to "Level 1" in NUREG/BR-0167, "Software Quality Assurance Program and Guidelines," February 1993, the primary guidance document for SAPHIRE, and to some software integrity level "1" activities in IEEE Std 1010-2004, "IEEE Standard for Software Verification and Validation. Releases of SAPHIRE will have to pass the acceptance tests given in the Acceptance Test Plan.

A *released* version of SAPHIRE represents an incremental version of the "current release" that is made generally available. Note that at times, significant enhancements and additions were introduced as part of these released versions, so while existing bugs may be fixed, it is possible that new bugs are introduced via these new features. Nonetheless, for each incremental version, the SAPHIRE software must pass an extensive automated test process to ensure that existing calculation features are not compromised. Definitions of the software release terms used by the SAPHIRE development team include:

Beta	The "beta" version of SAPHIRE is that numbered version (e.g., 8.x) that is currently under development at the INL. This version is used to add new features and to make significant modifications to either the analysis or user interface portions of the software. Since this version is in development, it is possible that features are incomplete or modification may leave the software in an unstable state. In addition, the software documentation may not be available specific to this version of the software. This version is not available for general release.
Current Release	The "current release" version of SAPHIRE is the most recent numbered version of the software that is "frozen." The term "frozen" indicates that the analysis and user interface portions of the software will not be modified, with the exception of needed changes related to programming errors or limitations. Typically, the current release is the version that undergoes the largest amount of use, and consequently, has the highest degree of testing.
N-1 Release	The "N-1 release" version of SAPHIRE is the second-to-last released "frozen" version.

Note that for all versions of SAPHIRE, transfer of the software or related information (in electronic or hardcopy format) is prohibited unless prior approval is obtained since the software is subject to U.S. export control regulations.

For the SAPHIRE QA, a variety of techniques is used to assure the integrity of the SAPHIRE software, including:

- Design changes
- Tests
- Documentation
- Version control
- Bug fixes

1.2.1 Project Scope and Organization

All NRC work assigned to a DOE national lab is governed by NRC Management Directive 11.7, NRC Procedures for Placement and Monitoring of Work with the Department of Energy. The NRC assigns each project a unique Job Code Number (JCN), is funded separately, and is

assigned a NRC Project Manager and NRC Technical Monitor. NRC Management Directive 11.7 establishes a controlled and monitored process for requesting services of a national lab, work planning, work authorization and initiation, work progress monitoring, reporting, work termination and project closeout.

The organizational structure of the SAPHIRE software development team influences and controls the software quality. Roles and responsibilities within the organizational structure provide the development team with the freedom, flexibility and objectivity to evaluate and monitor the software quality as well as verify problem resolutions. This structure enables the development team to tailor the maintenance and development activities, techniques, and methodologies for problem identification, reporting and resolution, testing, records retention, and configuration management.

For the INL, Software Quality Assurance (SQA) requirements are contract driven and interpreted from DOE Order 414.1C, “Quality Assurance”, 10 CFR 830 Subpart A, “Nuclear Safety Management”, and ASME NQA-1-2000, “Quality Assurance Requirements for Nuclear Facility Applications.”

INL will follow NRC Management Directive 11.7 “Procedures for Placement and Monitoring of Work with the Department of Energy” related to software development. This directive suggests that “all software development, modification, or maintenance tasks shall follow general guidance provided in NUREG/BR-0167 “Software Quality Assurance Program and Guidance.” SAPHIRE 8 will follow the requirements for Level 1 software defined in Section 1.2 of NUREG/BR-0167. The NRC performs an audit of the software QA implementation once a year against the requirement of NUREG/BR-0167.

1.2.2 Management

The organizational structure of the SAPHIRE software development team influences and controls the software quality. Roles and responsibilities within the organizational structure provide the development team with the freedom, flexibility and objectivity to evaluate and monitor the software quality as well as verify problem resolutions. This structure enables the development team to tailor the maintenance and development activities, techniques, and methodologies for problem identification, reporting and resolution, testing, records retention, and configuration management.

As SAPHIRE is currently in the operations and maintenance phase of the software development lifecycle, software development procedures and supporting standards are tailored to provide an appropriate level of quality, based upon a graded approach. The graded approach integrates the following INL software management processes, standard, and procedures:

- Software Management which identifies responsibilities, development methodologies, tools, and deliverables

- Quality Assurance activities to assure that the final software application meets the customer needs for quality and timeliness

- Configuration Management and Change Control to monitor and uniquely identify baselines, changes that are requested, evaluated, approved, and tested, as well as backup and recovery actions

- Software defect reporting and resolution for promptly addressing and resolving software errors

- Maintenance of the software to remove latent errors (corrective maintenance), respond to new or revised requirements (preventive maintenance), and to adapt to software changes in the operating environment (adaptive maintenance)

- Requirements and Design activities identified in contract documents

- Testing activities, including automated test scripts and results identified in the SAPHIRE Acceptance Test Plan. These test procedures demonstrate the adherence to the requirements specified in the NRC forms.

- Recording and implementing lessons learned

1.2.3 Tasks and Responsibilities

Management provides oversight activities as well as monthly status reports, draft reports, and a final report of the TV&V activities that are performed. The SAPHIRE principal investigator directs the roles, responsibilities, and tasks of the software development team. Many of the quality management tasks and activities are conducted by product teams but are also reviewed by the principal investigator.

1.2.4 Change Design and Testing Procedure

Software developers follow the SAPHIRE Change Design and Testing Procedure when adding a new feature or revising an existing capability. This procedure first describes the general approach to changes, and then describes processes that are more specific. The process stages include design and development, testing, and documentation. The initial design effort corresponds to the size and complexity of the change. Developers use a cyclical prototyping design methodology as a means to clarify and refine the change. The prototyping process involves the requestor throughout development. The developers will interact with the requestor(s) both initially and throughout the design and development process to ensure the change accomplishes the expected goal.

Changes and additions to the software vary from very small bug fixes to significant enhancements and new capabilities. The complexity of a change or addition also varies by item. Therefore, the developers use a graded approach to design. They spend more time and effort on larger and/or more complex changes than on relatively simple items. Areas of changes or bugs also dictate the level of effort. For example, problems in cut set generating are much more important than problems in report areas. Enhancements to cut set generation are researched much more carefully than enhancements to reports.

The frequency and formality of communications with the requestor also corresponds to the size and complexity of the change. This ensures that time and money is spent wisely.

The SAPHIRE developers utilize a cyclical, or whirlpool, prototyping software development methodology. The developers prepare prototypes of a proposed change or system, which can then be evaluated by both the developer and requestor, resulting in the development of a more refined prototype. This iteration process helps to clarify requirements, identify weak areas, and evolve and refine the design. Pictorially, the iteration process resembles a spiraling whirlpool or a target, where with each iteration, the cycle becomes smaller and tighter, until the final goal is achieved.

The cyclical prototyping methodology requires a starting point, which entails a reasonably clear definition of the initial problem and a general solution. When this has been achieved, the iterative development cycle begins.

The first step in designing a change to SAPHIRE requires that the developers and requestors define and discuss the problem and propose a solution. The developer should gain a broad understanding of the goal of the change, and the requestor should understand in general terms how the proposed solution will accomplish the goal.

At this point in the process, the change will be summarized in a SAPHIRE Change Request Form (see Appendix B), where the problem will be summarized and categorized. Once a clear definition of the change has been identified, additional items are considered, including:

- When applicable, define the necessary inputs and expected outputs.
- Determine the approximate complexity and level of effort required to accomplish the task.
- Consider how existing code functionality can be leveraged to help accomplish the task.
- Consider potential effects on other parts of SAPHIRE.

The next step is to prove the concept. This means developing key internal functions as well as a rudimentary interface to access and test those functions. This step serves to test the feasibility of the solution, and helps the designers understand the problem. The results of this step are used for further discussion between the developer and the requestor. This is considered the first iteration of the prototype. Depending upon the results, the design may be modified and refined. The prototype will be modified or rewritten to reflect the information learned.

An iteration of the software should improve the functionality of the change to bring it closer to its goal. Successive passes, as the design and prototype stabilize, will incorporate more and more of the following items:

- Additional supporting functions
- Refined and more complete user interface
- Integration into the SAPHIRE user interface
- Auxiliary functions to facilitate ease of use

Auxiliary functions are niceties that contribute to ease of use. They vary according to the task, but may generally include such things as customizing, sorting, and/or saving data, generating reports, loading and extracting data between projects, toolbar short-cuts, and individual and bulk processing of data. These types of auxiliary functions are added as time and budget permit. Depending on the scope and complexity of the task, the requestor and the developer maintain contact throughout the development process. Specifically, the requestor or a designated group of users will be given the opportunity to see, try, and comment upon prototypes at logical points.

As a prototype is refined, it approaches a point where satisfies the solution requirements. At this point, the SAPHIRE Change Design and Testing Checklist is completed. Completing this checklist will help assure that a standard list of coding issues have been addressed.

1.2.5 Approach to Bug Fixes and New Features

As new features and bug fixes are made, the INL developers follow a standard approach to integrating these items into SAPHIRE. For bug fixes, notes are taken from the reporting user describing the general context of the bug, as well as systematic actions to reproduce the bugs. This bug information includes acquiring a copy of the user's database, when necessary. Reporting problems or suggesting features can be done using the SAPHIRE web site (http://saphire.inl.gov) through the change request function. (See Appendix B for additional information)

A software problem is classified and prioritized according to severity. A bug is considered "minor" if it inconveniences the user, but a workaround exists to produce a correct answer. A bug is "major" if it prevents the user from obtaining the correct answer. Problems in more commonly used features are considered a higher priority than those found in less used features. User deadlines are also considered.

Bug fixes are tested in the environment in which they were reported, as well as other places if possible side effects are suspected. Sometimes, a release candidate is made available to the reporting user or group of users to ensure that the problem has been satisfactorily fixed. Once a bug has been resolved, it is added to the list of changes for the next official version, which must pass the set of acceptance tests described in the next section.

Software enhancements follow much the same approach as bug fixes. Enhancements are prioritized and implemented, with intermediate testing by the developer and often by the requestor. Once the process and results appear acceptable, the feature is added to the next official release.

1.2.6 Configuration Management and Control

Quality assurance reviews configuration management and control processes to ensure that only authorized changes are made to the software. All software modules that have been tested, documented, and approved for inclusion into the next release of the software are baselined. The software/system database "librarian" controls the baselined source code. Copies of current build routines needed to construct the software, including all copies of all build routines used in all prior releases are also under the librarian control.

SAPHIRE uses a configuration management database as a control library for all information related to the development of software fixes, enhances, baselines, and subsequent releases. Processes are in place to uniquely identify all components, modules, documentation, error reports, test suites, and test results through the establishment of a configuration control tracking number. The control library is kept on a server, where back-ups are regularly made. (Individual developers/programmers machines are periodically backed up as well). Controls are in place to preclude multiple users from simultaneously accessing the same information. A source code version control library requires that individual programmers “check-out” all files that they intend to modify. Prior to “check-in”, programmers must explain any changes made. A record is kept of all changes, both as explained by the developer, and as individual copies of each version of a file. At any time, the developer can retrieve past versions intact, if necessary. The SAPHIRE software program is continually modified, in response to user reported bugs and suggestions, and contractually specified enhancements. The version control procedure ensures a methodical approach to tracking and releasing these changes.

Bug fixes and all supporting documentation are placed under configuration control. Notes from the reporting user are obtained describing the general context of the bug, as well as step-by-step actions to reproduce the bugs. This includes acquiring a copy of the user’s database, when necessary. The bug is classified and prioritized according to severity. A bug is considered “minor” if it inconveniences the user, but a workaround exists to produce a correct answer. A bug is “major” if it prevents the user from obtaining the correct answer. Bugs found in more commonly used features are considered a higher priority than those found in less used features. User deadlines are also considered. Bug fixes are tested in the environment in which they were reported, as well as other places if possible side effects are suspected. Sometimes, a release candidate is made available to the reporting user or group of users to ensure that the problem has been satisfactorily fixed. Once a bug has been resolved, it is added to the list of changes for the next official version, which must pass the set of acceptance tests described in the next section.

Software enhancements and supporting requirements and documentation are also placed under configuration control. Enhancements are prioritized and implemented, with intermediate testing by the developer and often by the requestor. Once the process and results appear acceptable, the feature is added to the next official release.

SAPHIRE uses a configuration management database as a control library for all information related to the development of software fixes, enhances, baselines, and subsequent releases. Processes are in place to uniquely identify all components, modules, documentation, error reports, test suites, and test results through the establishment of a configuration control tracking number.

Bug fixes and all supporting documentation are placed under configuration control. Notes from the reporting user are obtained describing the general context of the bug, as well as step-by-step actions to reproduce the bugs. This includes acquiring a copy of the user’s database, when necessary. The bug is classified and prioritized according to severity. A bug is considered “minor” if it inconveniences the user, but a workaround exists to produce a correct answer. A bug is “major” if it prevents the user from obtaining the correct answer. Bugs found in more commonly used features are considered a higher priority than those found in less used

features. User deadlines are also considered. Bug fixes are tested in the environment in which they were reported, as well as other places if possible side effects are suspected. Sometimes, a release candidate is made available to the reporting user or group of users to ensure that the problem has been satisfactorily fixed. Once a bug has been resolved, it is added to the list of changes for the next official version, which must pass the set of acceptance tests described in the next section.

Software enhancements and supporting requirements and documentation are also placed under configuration control. Enhancements are prioritized and implemented, with intermediate testing by the developer and often by the requestor. Once the process and results appear acceptable, the feature is added to the next official release.

1.2.7 Acceptance Testing/Automated Testing

Prior to any official SAPHIRE release of Version 8, the software is run through a series of automated tests. The tests simulate user input to the computer through a test script, and results are captured and compared to expected results. This ensures that given a static input PRA file, the risk or reliability results from SAPHIRE will be consistent from one release to the next.

These tests were developed by first identifying the critical tasks performed in a PRA. Then these tasks were mapped to the SAPHIRE functions that perform these tasks (Appendix C contains additional detail). The critical functions were determined to include the following:

- Fault tree analysis
- Event tree and sequence analysis
- End state analysis
- Importance measures analysis
- Uncertainty analysis
- Change sets
- Data utility functions
- Workspace functionality

Next, a variety of models are selected, with varying degrees of size and complexity, based on suitability for adequately testing one or more critical functions. These models mainly consist of actual PRA models developed by experienced analysts.

Test scripts were developed to exercise essential SAPHIRE functions, with a quantitative emphasis. The test scripts mimic actions taken by an analyst, such as starting SAPHIRE and navigating the user interface by selecting menu options, clicking buttons and typing information. Results are saved and compared against expected results. A summary and a detailed report of the results of the tests are produced, so that an overview of the results can quickly be determined, and any failures (or successes) can be traced in more detail.

A change is not considered complete until the results have been tested and found reasonable. Developers and key users will test to see that the change works as expected and is free of defects. Changes and new capabilities will not be released until the results are deemed

satisfactory and correct. When the change has been accepted, the SAPHIRE Change Form will be updated to document the completion of development.

Prior to official release of a version, SAPHIRE's automated test suite must complete successfully (100% of all tests). The success of the suite is a good indicator that the new change does not adversely affect other areas of the code. Rarely do changes and bug fixes change the acceptable results of the test. On the unusual occasion when this happens, the target test results are modified to match the new accepted results for future runs. The reasons for the results modification are documented and cleared by an authority on the subject matter.

When the tests produce expected results, the correctness and stability of SAPHIRE is validated. The tests exercise various features on assorted databases, with substantial overlap on key features to provide added confidence.

Quality is not “built-in” through the testing process, rather, quality is implemented throughout the lifecycle, beginning with the examination of the requirements, design, lessons learned from previous releases and reviews of software defect reports.

A Software Verification and Validation Plan (SVVP) was developed to make sure that all requirements specifications in the requirements traceability matrix (RTM) are implemented and those new features do not affect existing code functionality or design. The SVVP is a consolidated document used in conjunction with the RTM and design documentation, for tracking the software development, testing and implementation. Testing of new features implemented for each release of the software is done in accordance with the Acceptance Test Plan. The SVVP, ATP, and design documentation is updated, as necessary, for each release of SAPHIRE by the development team.

Models, with varying degrees of size and complexity, based on suitability for adequately testing one or more critical functions are then selected. These models mainly consist of actual PRA models developed by experienced analysts. Test scripts have been developed to exercise essential SAPHIRE functions, with a quantitative emphasis. New test scripts are developed for software enhancements, as needed. These test scripts mimic actions taken by an analyst, such as starting SAPHIRE and navigating the user interface by selecting menu options, clicking buttons and typing information. Results are saved and compared against expected results. A summary and a detailed report of the results of the tests are produced, so that an overview of the results can quickly be determined, and any failures (or successes) can be traced in more detail.

The test suite is evaluated against significant changes and new features. New tests are developed to check a new feature when the developer and customer agree that it is appropriate. To develop a new test, a suitable test scenario with a database and validated correct answers must be determined.

SAPHIRE testing also utilizes a test witness monitor form as shown in Figure 2.

SAPHIRE Test Witness Monitor Form

Name of Witness:	
Signature of Witness:	
Date:	
Time:	
Software Version being tested:	
(Name and Build Number Version)	
Software Test Procedure(s) used:	
(Name and Version)	
Test Platform(s) Used:	
Test type:	Automated [ ] Manual [ ]
Test Results:	Pass [ ] Fail [ ]

PRE-TEST ACTIVITIES	**STATUS**
Test Schedule Established	YES [ ] No [ ] N/A [ ]
Test Procedures Reviewed	YES [ ] No [ ] N/A [ ]
Test Environment setup in accordance with Test Procedures	YES [ ] No [ ] N/A [ ]
Test Activities	
Test Procedures are used	YES [ ] No [ ] N/A [ ]
Test Environment used	YES [ ] No [ ] N/A [ ]
Test Deviations documented	YES [ ] No [ ] N/A [ ]
Test Nonconformance documented	YES [ ] No [ ] N/A [ ]
Post Test Activities	
Verify Test Deviations Documented	YES [ ] No [ ] N/A [ ]
Test Deviations Corrected	YES [ ] No [ ] N/A [ ]
Additional Comments	

Figure 2. SAPHIRE Test Witness Monitor Form

Each new version of SAPHIRE undergoes NRC review and beta testing before its release as shown in the release management flow diagram in Figure 3. Beta testing helps to ensure that the results produced by the new version are correct and that the software is user-friendly and functional. Beta testers are analysts experienced with PRA methods and terminology and typically are familiar with earlier versions of SAPHIRE.

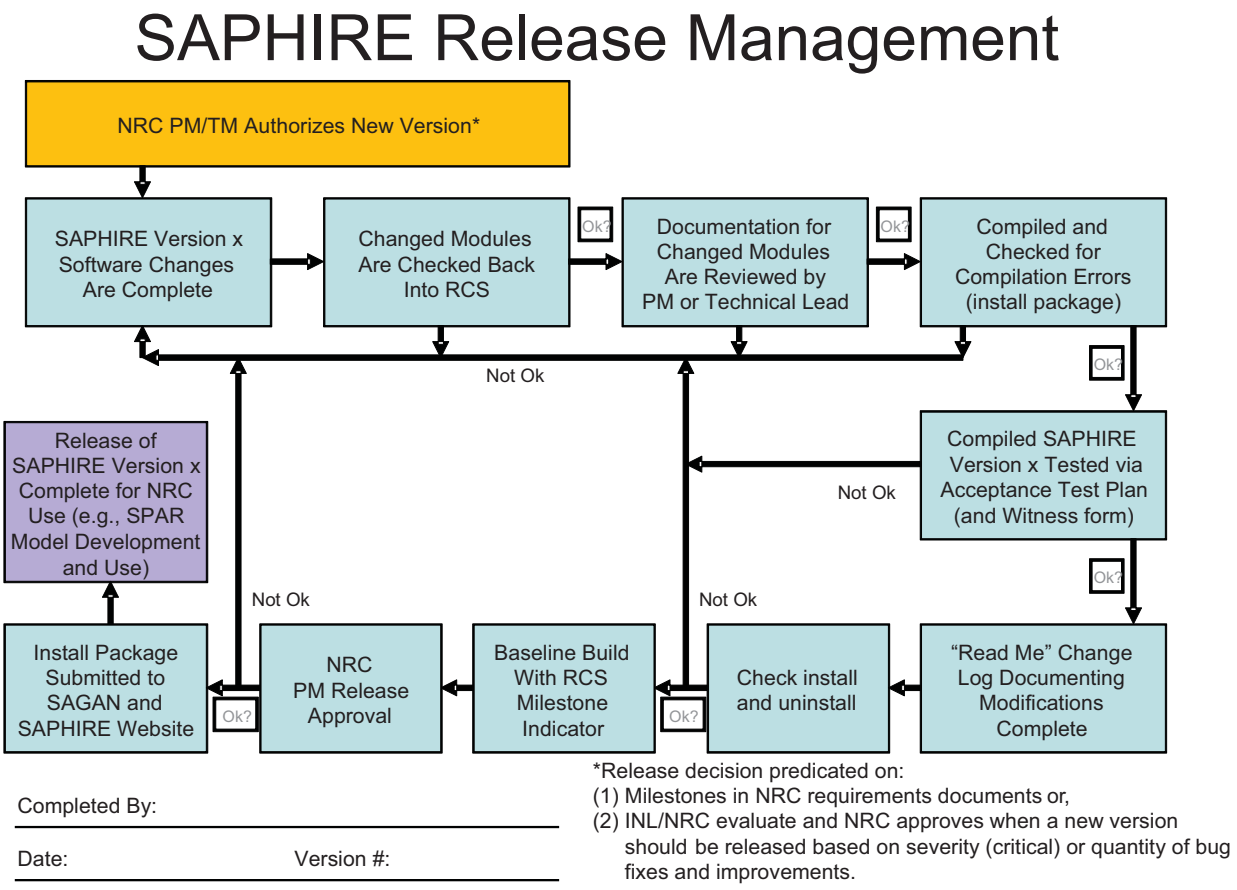


Figure 3. SAPHIRE release management process

In addition to the automated testing employed by the SAPHIRE quality assurance, the development team utilizes a multi-faceted approach to testing. This approach, illustrated in Figure 4, is comprised of three items: internal testing, external testing, and automated testing. "Internal" testing (or developmental testing) includes those checks performed by the development team itself to ensure quality during the development process. External testing are those evaluations performed by risk and reliability end-users using, in many cases, "real world" models. Lastly, the automated testing are those tests that are used to ensure quality for each incremental SAPHIRE release and are described in Appendix C of this report.

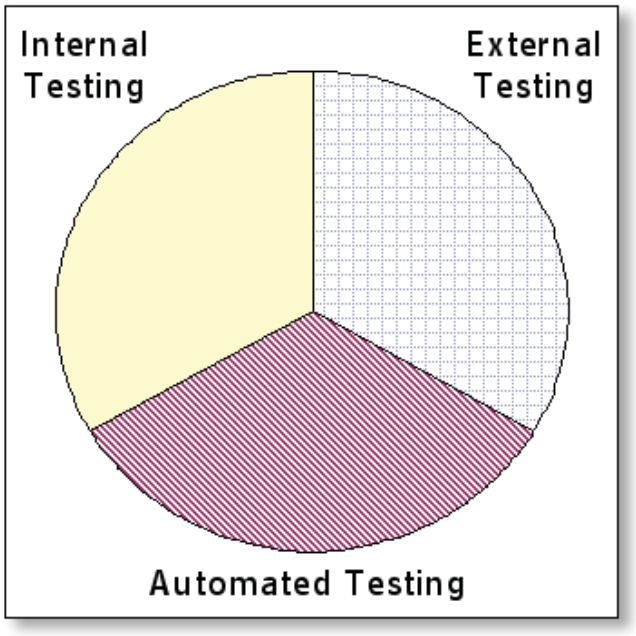


Figure 4. Types of testing used during the SAPHIRE development process

1.3 Reviews

As part of the SAPHIRE 8 development, the team will perform and document, as appropriate, periodic peer reviews and code walkthroughs, including reviews of preliminary and critical designs proposals. Since non-conformances are reported and logged through the SAPHIRE Change Request tracking system, medium and high priority reports will be reviewed (and the review documented).

Peer reviews and code walkthroughs will be reviewed independently when possible by IV&V members. NRC performed an internal peer review, and not an external peer review for the first release of SAPHIRE 8. The NRC SAPHIRE users also test and comment on SAPHIRE code modifications. In addition, NRC performs audits against NUREG/BR-0167 for SAPHIRE.

Reviews that have been completed include:

- Preliminary design review by INL on SDP interface
- Preliminary design review by INL human factors department
- Preliminary design review by INL
- Preliminary design review by the NRC on SAPHIRE

- Preliminary design review by the NRC on the SDP module
- Design review by the NRC as part of the internal peer review
- Design review by INL

To help in design of the software, INL has also consulted experts within INL (e.g., human factors experts in the design of a user interface).

The NRC also performs periodic reviews of the SAPHIRE quality assurance against the guidance in NUREG/BR-0167, “Software Quality Assurance Program and Guidelines,” February 1993.

1.4 Independent Verification and Validation

The purpose of the Independent Verification and Validation (IV&V) role in the evaluation of the SAPHIRE development is to assess the activities that results in the specification, documentation, and review of the requirements that the software product must satisfy, including functionality, performance, design constraints, attributes and external interfaces. The IV&V for SAPHIRE 8 started after the software engineering and software development of SAPHIRE had already been in development. Consequently, the IV&V reviewed the requirements specified in the NRC requirements documents to verify these requirements were being followed. As part of the IV&V, the traceability of requirements is of concern. Requirements traceability is essential to all software development activities. Without a well documented way to trace requirements, design components may be overlooked, and test cases missed.

For the IV&V team to properly evaluate the requirements, it had to obtain requirements from the NRC requirements documents. In addition, the NUREG/BR-0167, Software Quality Assurance Program and Guidelines, requires the development of Software Requirements Documentation that specifies the requirements that the software to be developed/maintained must meet. 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.

The IV&V team provided status, interim reports, and a final report to the SAPHIRE development team. The IV&V Software Requirements Documentation is intended to provide the specification, documentation, and review of the requirements to meet the contractual commitments prepared by the sponsor, the Nuclear Regulatory Commission. Further, IV&V evaluates and assesses the processes and products developed during each phase of the software life cycle. 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 QA 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. Feedback from the IV&V team was used to improve the quality of SAPHIRE 8 as it neared its final release.

1.4.1 Version Control

The INL software developers use version control for both the formally released SAPHIRE versions, as well as for source code. For each formal release of the software, the developers perform an acceptance test: the software must pass a suite of automated tests prior to official release.

Each official release of SAPHIRE is assigned a unique version identifier. The release is bundled into a standard installation package for easy and consistent set-up by individual users. Included in the release is a list of bug fixes and new features for the current release, as well as a history of those items for past releases. Each formal release of SAPHIRE will have passed an acceptance test described in the Acceptance Test Plan provided in Appendix C (the Acceptance Test Plan, however, will be updated as necessary).

In addition to assignment of a unique version identifier for an official software release, each source code file is kept in a controlled library. (Source code is a collection of all the computer instructions written by developers to create the finished product.) The library is kept on a server, where back-ups are regularly made. (Individual developers/programmers machines are periodically backed up as well.)

The source code version control library requires that individual programmers "check-out" all files that they intend to modify. Prior to "check-in", programmers must explain any changes made. A record is kept of all changes, both as explained by the developer, and as individual copies of each version of a file. At any time, the developer can retrieve past versions intact, if necessary.

The SAPHIRE software program is continually modified, in response to user reported bugs and suggestions, and contractually specified enhancements. The version control procedure described above ensures a methodical approach to tracking and releasing these changes.

1.4.2 QA Standards and Practices

Since INL follows NRC Management Directive 11.7 “Procedures for Placement and Monitoring of Work with the Department of Energy” related to software development, INL also follows general guidance provided in NUREG/BR-0167 “Software Quality Assurance Program and Guidance.” SAPHIRE 8 will follow the requirements for Level 1 software defined in Section 1.2 of NUREG/BR-0167.

The content of all QA standards, processes and procedures as well as documentation and coding conventions that are utilized are assessed to ensure the quality of the SAPHIRE code and supporting information used to construct the software release. Quality functions include the reviews of the basic design and programming activities involved. Information under the cognizance of the quality review includes, but is not limited to the following:

- Documentation standards
- Design standards
- Coding standards
- Commenting standards
- Testing standards

To assess these items, QA reviews of software requirement specifications, design specifications, verification and validation plans, test documentation, and configuration management processes. Methods used to assess these items include functional audits to ensure that all requirements are being implemented, physical audits to verify the consistency, completeness, and correctness of the software, software documentation and its readiness for release, and in-process audits to verify the consistency of the design.

Many of these activities for SAPHIRE are conducted as identified in the Software Acceptance Test Plan. This includes reviews of the contract documents, which provide the basic requirements for maintaining the SAPHIRE software. As stated above, the development team conducts automated testing to assure that all requirements have been implemented correctly.

Metrics are an integral part of tracking quality of the software development process and products. While numerous possible metrics exist, the SAPHIRE project will focus on using the metrics of:

- Earned Value to track cost and schedule variances. This metric as measured by variances will be reported monthly to the NRC.

- Trending of calculation errors as reported in the Change Request tracking system.

Other metrics derived from source code, such as McCabe's complexity measure, may be considered for future use.

In addition to formal reviews, INL also conducts code review and inspections when legacy code is ported over to the new development environment.

1.4.3 Documentation

As changes to SAPHIRE are finalized, a description of the change is documented in several places. The developers describe the change when they check-in the altered source code into the version control library. Upon official release, the change is noted in a "read me" text file that is distributed with SAPHIRE.

SAPHIRE has a help reference and training manuals, including one specific for the SAPHIRE Version 8 user interface "Significance Determination Process." These documents are maintained as necessary to reflect new features and capabilities.

Documentation is traditionally developed and implemented to govern and provide quality assurance oversight of the requirements implementation, product design, code development and testing, verification, validation and maintenance of software. As the SAPHIRE product is

currently in an operations and maintenance mode, the focus is primarily on providing enhancements and minor bug fixes. As such, a graded approach is applied to provide a tailored method for document generation. The development team obtains and retains change request information and documents lessons learned from previous development efforts. Materials for new releases are developed to provide the end user with documents that identify the SAPHIRE product's key functional area, the cut-solving algorithm. These documents provide the mechanism for the product team to perform internal quality reviews to ensure that all requirements for product enhancement and/or bug fixes have been implemented.

User documentation includes the SAPHIRE Advanced Training Manual, the SAPHIRE User's Manual, and the SAPHIRE Technical Reference Manual. These manuals are updated as necessary to reflect changes in the software.

Each release of SAPHIRE is bundled into a standard installation package for easy and consistent set-up by individual users. Included in the release is a list of bug fixes and new features for the current release, as well as a history of those items for past releases.

2. TESTING APPROACH

The test approach used for SAPHIRE Version 8 will be based upon the test approach for previous releases of the SAPHIRE tool. Taking into consideration lessons learned from the SAPHIRE V&V efforts, where applicable, actual tests and test specifications from the older testing were used. Additional tests were developed specifically for the newer QA process, primarily due to the fact that the test could be automated. This automation aspect of testing allows the testing team to rerun a battery of calculations as many times as they wish, regardless of the complexity of the test. In order to decide which tests were to be used and why, a test plan was developed. This plan followed the general procedures used in the earlier V&V efforts, but was modified to take advantage of unique features found when performing automated testing. Thus, the updated testing plan for the QA includes the following steps:

- Determining the areas requiring testing. This step is similar to the V&V process of identifying vital and non-vital functions. Additional features are checked in the current V&V process then were tested in the previous V&V efforts.

- Developing the test model, including the identification of available SAPHIRE PRA databases that would adequately test SAPHIRE functions.

Availability of a variety of different plant data models enhances the viability of the test suite. Core features are exercised repeatedly across tests (and their associated models) in the process of performing each test's specific analysis task. Use of different plant and database models, from the simple DEMO database to the SPAR Revision 2QA, SPAR 3i models to NUREG-1150 models, provides quality and reliability assurance that any variations among models are appropriately handled by any released version of SAPHIRE. While the current tests do not address every feature within SAPHIRE, they do cover the important calculational parts of the software. Also, some specific PRA areas are tested using only a single test case.

Models and test cases are added as needed to the test suite to improve the overall coverage of testing vital functions in SAPHIRE.

2.1 Features to be tested

To determine the SAPHIRE features most important to be tested, the critical tasks performed in a PRA (e.g., fault tree analysis, event tree analysis, sensitivity analysis) were identified. Then the SAPHIRE functions needed to accomplish each of these tasks were determined. The review process produces a list of items to be tested, which PRA analysis experts using SAPHIRE reviewed and revised. In summary, the following SAPHIRE functions are tested:

- Fault Tree Analysis, including cut set generation and quantification, application of recovery rules (i.e., modifications made to the cut set results after they are generated), and the capability to perform the analysis on a single fault tree or on multiple fault trees

- Event Tree and Sequence Analysis, including event tree sequence generation, sequence cut set generation, quantification, application of recovery rules, application of partition rules (i.e., steps to move particular cut sets to a specified end state category), and the capability of performing the analysis on a single event tree/sequence or on multiple event trees/sequences

- End State Analysis, including gathering cut sets by sequence end-state designation, gathering of cut sets by partitioning rules, quantification, and the capability of performing the analysis on a single end state or on multiple end states

- Importance Measures Analysis, for options available to quantify importance measures

- Uncertainty Analysis, for individual fault trees, using either Latin Hypercube or the Monte Carlo sampling methods

- Uncertainty Analysis, for individual sequences or groups of sequences, using either Latin Hypercube or the Monte Carlo sampling methods

- Change Sets, and other similar features, providing the capability to perform sensitivity analyses (change sets contain user-defined modifications to basic event probabilities)

- Data Utility Functions intended to facilitate data handling and manipulation

- Mapping of systems, components, failure modes to basic events and then using them in sequence analysis

- Linking event trees with the “Generate cut set” option checked used for the “large event tree” PRA methodology

- Transformations, which are generally used only for fire or flooding analyses. Note that version 8 does not currently allow for editing transformations, but they can be tested and called through macros

- Seismic analysis

- Stress testing (record count up to 2 million sequences)

- Verification of the new user interface

- Significance Determination Process , Event and Condition Analysis, and General Analysis workspaces

- External Event Model Results

- Shutdown Model Results

To test the above SAPHIRE functions, a variety of models were selected, with varying degrees of size and complexity, based on their suitability for adequately testing the selected functions.

The intent of this effort was to acquire basic assurance that new updates or changes have not compromised any existing capabilities. Size and boundary conditions of the PRA models were not major issues. Databases of typical size and complexity were selected from among the available models. A decision was made to test more features with less complex models than to test fewer features with complex models. Generally, actual PRA models developed by experienced analysts for analyzing nuclear power plants were used for the tests. Several tests, including uncertainty distributions, importance measures and change sets were conducted on artificial plant models. At a minimum, each feature tested was evaluated with at least two PRA models. Further, many of the basic features (e.g., basic event probability generation, and minimal cut set solving) were tested by almost all the PRA models owing to the need to perform these basic functions as part of a more complex calculation.

2.2 Features Not Tested

Like most software-development projects, time and budget constraints prohibited exhaustive testing. The verification effort focused on quantitative aspects of SAPHIRE. While the tests and acceptance criteria address a large part of the calculative functionality within SAPHIRE, the tests do not cover 100% of SAPHIRE capabilities. For example, the current test suite did not encompass every possible way of modifying cut sets after generation. Users can manipulate cut sets after generation (e.g., "post-processing") by manually editing them, using "recovery rules," using the "prune" option, and performing a cut set update. However, if a modification is made to SAPHIRE, for example, to update the recovery rule algorithm, the existing test suite will ensure the modification did not change test results through regression testing. But the test suite does test the most commonly used mechanisms of performing tasks in SAPHIRE. Other calculative aspects not tested include the following:

- Event probability cut off (not frequently used due to the calculation speed of modern computers and software such as SAPHIRE)
- Solving sequences without fault trees (an obsolete calculation technique that may be removed from future versions of the software)
- Starting gate name (generally used only during development)

2.3 Test Descriptions

This section contains a complete list of test descriptions, referenced by one or more test scenarios in the report. Note that, unless otherwise specified, SAPHIRE Version 8 test results were compared with results from SAPHIRE Version 7 test results. Each test may be performed on multiple PRA models.

Workspaces provide areas to analyze working copies of the models without affecting the models themselves. There are three workspaces that provide different functionalities, Significance

Determination Process (SDP), Events and Condition Assessment (ECA), and General Analysis (GA). If the workspace is not noted in the test description, the test is actually impacting the baseline model. The interaction with the baseline model is usually defined as the Standard Analysis interface. Note, for each test below, the Standard Analysis interface is tested unless a specific Workspace (e.g., Events and Condition Assessment) is specified in the test title.

2.3.1.1 Test-01, Solve Fault Trees

Scenarios generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify fault tree minimal cut sets, and recovery rules. The alternate case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each fault tree.

2.3.1.2 Test-02, Core Damage Frequency

Scenarios generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The alternate case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.

2.3.1.3 Test-03, Events and Condition Assessment: Auxiliary Feed Water out of service for 72 hours

Scenarios exercise all aspects of operational event analysis including removal of equipment from service and automated processing of all steps. These steps include basic event generation with change sets; and generation, quantification, and recovery of cut sets. The number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified.

2.3.1.4 Test-04, Events and Condition Assessment: Emergency Diesel Generator out of service for three months

Scenarios exercise all aspects of operational event analysis, including removal of equipment from service and automated processing of all steps. These steps include basic event generation with change sets, and generation, quantification, and recovery of cut sets. The number of sequences, total CCDP, total core damage probability (CDP), total importance, and CCDP, CDP, and importance for each sequence are verified.

2.3.1.5 Test-05, Initiating Event Assessment: Transient with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.6 Test-06, Initiating Event Assessment: Small Loss of Coolant Accident (SLOCA) with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated

steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.7 Test-07, Initiating Event Assessment: Steam Generator Tube Rupture with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.8 Test-08, Initiating Event Assessment: Grid-Related Loss of Off-Site Power (LOOP) with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.9 Test-09, Initiating Event Assessment: Plant-Centered LOOP with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.10 Test-10, Initiating Event Assessment: Severe Weather LOOP with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.11 Test-11, Initiating Event Assessment: Extreme Severe Weather LOOP with no other failures

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.12 Test-12, Initiating Event Assessment: Transient with AFW Failed

Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated

steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.

2.3.1.13 Test-13, Dominant Sequence Frequencies and Core Damage Frequency Uncertainty

This scenario continues the tracking with an automated test script. Cut sets generated with cut set probability cutoff and cut set size cutoff. Recovery rules are applied without cutoff. Cut set update performed with no truncation. Project level Monte Carlo uncertainty performed on results using 5000 samples.

2.3.1.14 Test-14, Fault Tree Uncertainty: Monte Carlo Method/Log Normal Distribution

This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the log normal distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and error factors. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples (simulated values) and a random number seed of 4,321 for each test.

2.3.1.15 Test-15, Fault Tree Uncertainty: Monte Carlo Method/Normal Distribution

This scenario consists of variations that test uncertainty using the Monte Carlo simulation technique for the normal distribution type. Two fault trees are used that consist of an OR gate with a single basic event as its input, with differing basic event nominal probabilities and standard deviation values. Fault tree combinations of five sample sizes and two seed values are used for a total of ten tests for each tree. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.

2.3.1.16 Test-16, Fault Tree Uncertainty: Monte Carlo Method/Beta Distribution

This scenario consists of ten variations that test uncertainty using the Monte Carlo simulation technique for the beta distribution type. The ten variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.17 Test-17, Fault Tree Uncertainty: Monte Carlo Method/Chi Squared Distribution

This scenario consists of twelve variations that test uncertainty using the Monte Carlo simulation technique for the chi-square distribution type. For ten of the variations, ten fault trees are used that consists of an OR gate with a single basic event as its input. Each basic event has a different nominal probability and uncertainty value (degrees of freedom). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test. For the other variations two fault trees are used that consist of an OR gate with a single basic event as its input with differing basic event nominal probabilities and uncertainty values. For each of these fault trees, four different sample sizes

and seed of 4,321 are used. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.

2.3.1.18 Test-18, Fault Tree Uncertainty: Monte Carlo Method/Exponential Distribution

This scenario consists of eight variations that test uncertainty using the Monte Carlo simulation technique for the exponential distribution type. The eight variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.19 Test-19, Fault Tree Uncertainty: Monte Carlo Method/Uniform Distribution

This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the uniform distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.20 Test-20, Fault Tree Uncertainty: Monte Carlo Method/Gamma Distribution

This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the gamma distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values (r). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.21 Test-21, Fault Tree Uncertainty: Monte Carlo Method/Maximum Entropy Distribution

This scenario consists of seven variations that test uncertainty using the Monte Carlo simulation technique for the maximum entropy distribution type. The seven variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end and lower end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.22 Test-22, Sequence Uncertainty: Monte Carlo Method / Dirichlet Distribution

This test scenario consists of four variations that test uncertainty analyses using the Monte Carlo simulation technique for the Dirichlet distribution type. The first three variations each use a three-branch event tree with differing failure probabilities and parameter values. The fourth variation uses a 121-branch event tree. Change sets are used to correlate the basic events. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.

2.3.1.23 Test-23, Fault Tree Uncertainty: Monte Carlo Method/Seismic Distribution

This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the seismic distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event median failure acceleration, screening G-level, Beta-R and Beta-U values. Uncertainty analysis is performed using the Seismic analysis type. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 samples and a seed of 4,321 for each test.

2.3.1.24 Test-24, Fault Tree and Sequence Uncertainty: Monte Carlo Method/Constrained Noninformative Distribution

This scenario consists of five variations that test uncertainty using the Monte Carlo simulation techniques for the Constrained Noninformative distribution type. The three variations involving fault trees use fault trees that consists of an OR gate with a single basic event as its input with differing basic event nominal probabilities. The two variations involving sequences use event trees with differing initiating event nominal frequencies. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 simulated values for each test.

2.3.1.25 Test-25, Fault Tree Uncertainty: Latin Hypercube Method/Log Normal Distribution

This scenario consists of six variations that test uncertainty using the Latin Hypercube simulation technique for the log normal distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and error factors. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples (simulated values) and a random number seed of 4,321 for each test.

2.3.1.26 Test-26, Fault Tree Uncertainty: Latin Hypercube Method/Normal Distribution

This scenario consists of variations that test uncertainty using the Latin Hypercube simulation technique for the normal distribution type. Two fault trees are used that consist of an OR gate with a single basic event as its input, with differing basic event nominal probabilities and standard deviation values. Fault tree combinations of five sample sizes and two seed values are used for a total of ten tests for each tree. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.

2.3.1.27 Test-27, Fault Tree Uncertainty: Latin Hypercube Method/Beta Distribution

This scenario consists of ten variations that test uncertainty using the Monte Carlo simulation technique for the beta distribution type. The ten variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.28 Test-28, Fault Tree Uncertainty: Latin Hypercube Method/Chi Squared Distribution

This scenario consists of twelve variations that test uncertainty using the Monte Carlo simulation technique for the chi-square distribution type. For ten of the variations, ten fault trees are used that consists of an OR gate with a single basic event as its input. Each basic event has a different nominal probability and uncertainty value (degrees of freedom). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test. For the other variations two fault trees are used that consist of an OR gate with a single basic event as its input with differing basic event nominal probabilities and uncertainty values. For each of these fault trees, four different sample sizes and seed of 4,321 are used. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.

2.3.1.29 Test-29, Fault Tree Uncertainty: Latin Hypercube Method/Exponential Distribution

This scenario consists of eight variations that test uncertainty using the Monte Carlo simulation technique for the exponential distribution type. The eight variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.30 Test-30, Fault Tree Uncertainty: Latin Hypercube Method/Uniform Distribution

This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the uniform distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.31 Test-31, Fault Tree Uncertainty: Latin Hypercube Method/Gamma Distribution

This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the gamma distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values (r). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.32 Test-32, Sequence Uncertainty: Latin Hypercube Method/Maximum Entropy Distribution

This scenario consists of seven variations that test uncertainty using the Monte Carlo simulation technique for the maximum entropy distribution type. The seven variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end and lower end uncertainty values. The 5th percentile,

50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.33 Test-33, Sequence Uncertainty: Latin Hypercube Method/Dirichlet Distribution

This test scenario consists of four variations that test uncertainty analyses using the Monte Carlo simulation technique for the Dirichlet distribution type. The first three variations each use a three-branch event tree with differing failure probabilities and parameter values. The fourth variation uses a 121-branch event tree. Change sets are used to correlate the basic events. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified. Since this distribution type was not available in version 5, version 6 results have been inspected for acceptance and are used for comparison against subsequent incremental releases.

2.3.1.34 Test-34, Fault Tree Uncertainty: Latin Hypercube Method/Seismic Distribution

This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the seismic distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event median failure acceleration, screening G-level, Beta-R and Beta-U values. Uncertainty analysis is performed using the Seismic analysis type. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 samples and a seed of 4,321 for each test.

2.3.1.35 Test-35, Fault Tree and Sequence Uncertainty: Latin Hypercube Method/ Constrained Noninformative Distribution

This scenario consists of five variations that test uncertainty using the Monte Carlo simulation techniques for the Constrained Noninformative distribution type. The three variations involving fault trees use fault trees that consists of an OR gate with a single basic event as its input with differing basic event nominal probabilities. The two variations involving sequences use event trees with differing initiating event nominal frequencies. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 simulated values for each test.

2.3.1.36 Test-36, Fault Tree Uncertainty: Monte Carlo Method/Histogram Distribution

This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the histogram distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and histograms (of percentage, area, and range types). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.37 Test-37, Fault Tree Uncertainty: Latin Hypercube Method/Histogram Distribution

This scenario consists of four variations that test uncertainty using the Latin Hypercube simulation technique for the histogram distribution type. The four variations use fault trees that

consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and histograms (of percentage, area, and range types). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.

2.3.1.38 Test-38, Gathering of End States

This scenario generates basic event data (with no change sets) and gathers the end states (without cut set probability cutoff, by sequence end state). The alternate case min-cut upper bound and the number of cut sets are verified for each end state.

2.3.1.39 Test-39, End State Uncertainty: Monte Carlo Method

These scenarios perform multiple event sampling on all sequences that belong to a particular end state (single uncertainty), as well as the collection of all end states (group uncertainty). The mean, 5th percentile, median, 95th percentile, and standard deviation results are verified based on 3,000 simulated values for each test.

2.3.1.40 Test-40, End State Uncertainty: Latin Hypercube Method

These scenarios perform multiple event sampling on all sequences that belong to a particular end state (single uncertainty), as well as the collection of all end states (group uncertainty) . The mean, 5th percentile, median, 95th percentile, and standard deviation results are verified based on 3,000 simulated values for each test.

2.3.1.41 Test-41, Cut Set Verification

This test case consists of scenarios that compare cut sets from selected fault trees, sequences, and end states. The cut set frequency, percent contribution to the total, and basic events in the cut set are verified. Cut sets are solved and /or gathered with truncation, auto-recovered, and updated. Sequences and fault trees are solved with and without their default flag sets. Also, fault tree editing is briefly tested. This is done by opening the alphanumeric logic editor, saving and converting logic to graphics, then pulling up the graphical editor and saving the graphics. This test does not test specific editing features but it does verify that the original logic is correctly loaded and saved. Failure of the logic to be preserved correctly would be detected with incorrect cut set results.

2.3.1.42 Test-42, Link Small Event Tree

This scenario uses the Surry Large Early Release Frequency (LERF) Level 2/3 model (S_LERF) to link event trees using the small event tree methodology. Prior to link, each event tree is loaded into the graphical editor and saved to ensure that the correct logic is preserved. The sequences are then solved with cutoff. The alternate case min cut upper bound and number of cut sets is verified for each Level 1 sequence.

2.3.1.43 Test-43, Partition Sequence Cut Sets

This scenario applies event tree partition rules to the sequences generated in scenario reference number Test-42. These partition rules assign Plant Damage States (PDSs) to all sequences with cut sets. These end states are then gathered by cut set partition. The alternate case min cut upper bound and number of cut sets is verified for each PDS.

2.3.1.44 Test-44, Link Large Event Tree

This scenario uses the results from scenario reference number Test-43. The PDS event trees created by the partition rules are linked using the large event tree methodology and create sequence logic cut sets. The LERF end states are then gathered by sequence end state and re-quantified using the Rare Event approximation. The alternate case min-cut upper bound and number of cut sets are verified for each LERF end state.

2.3.1.45 Test-45, Fault Tree Importance Measures

This test case consists of scenarios that test importance measures calculations with fault trees for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.

2.3.1.46 Test-46, Sequence Importance Measures

This test case consists of scenarios that test Sequence importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.

2.3.1.47 Test-47, Sequence Group Importance Measures

This test case consists of scenarios that test Sequence Group importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.

2.3.1.48 Test-48, End State Importance Measures

This test case consists of scenarios that test End State importance measure calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.

2.3.1.49 Test-49, End State Group Importance

This test case consists of scenarios that test End State Group importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.

2.3.1.50 Test-50, Change Set Processing: Single

This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, single basic event changes are made in a change set. The change set is then marked and the basic event data is generated. An affected sequence is then selected and cut set results are verified.

2.3.1.51 Test-51 Change Set Processing: Class

This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, class basic event changes are made in a change set. The change set is then marked and the basic event data is generated. An affected sequence is then selected and cut set results are verified.

2.3.1.52 Test-52, Change Set Processing: Marked Order

This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, the change sets created in Test-50 and Test-51 are used. Multiple change sets are marked and the basic event data is generated. An affected sequence is then selected and cut set results are validated. This test verifies that the changed basic events are processed correctly based on the marked order of the change sets.

2.3.1.53 Test-53, Extract, Delete, Load, Solve: Fault Trees and Basic Events

This test consists of scenarios that exercise utility functions associated with the database for loading plant models, end state data or other information to be analyzed with the tool set.

2.3.1.54 Test-54, Fault Tree Utility Functions: Auto page, Solve, Save Cut Sets to End States

SAPHIRE provides several utilities maintain fault trees. These tests verify that the use of these features does not introduce errors into the database. The auto-page scenario breaks up a large fault tree into manageable smaller fault trees with transfer information. An auto-page is performed on a large fault tree, and then the modified tree is solved to verify the cut set results are not altered with the paging operation. Another scenario copies a fault tree cut sets to an end state, and then verifies that the cut sets in the end state match the cut sets in the fault tree.

2.3.1.55 Test-55, Event Tree Linkage (including rules)

The event tree linking rules are tested using several different databases. The databases are the Surry LERF model, Wolf Creek Revision 302, and Peach Bottom Revision 302. The Surry LERF model links the Level 1 event tree sequences together prior to solving the accident sequences, then performs an end state gather. The end states then become Level 2 event trees, which are linked together using the large event tree method. These Level 2 sequences are then gathered into the final end states for LERF, NO-LERF, etc. The Wolf Creek and Peach Bottom models have no accident sequences at the beginning. The test has the sequences being generated using dynamic flag sets for the accident sequences, and then evaluates the sequences. The sequences are evaluated using the developed dynamic flag sets and then with no flag sets.

2.3.1.56 Test-56, End-State Gathering

The end state gathering process is tested using the Surry LERF model and the Beaver Valley NUREG 1150 model. Both models have the sequences gathered into end states. The Surry LERF model uses partition rules, while the Beaver Valley model uses the end state name.

2.3.1.57 Test-57, Compound Event Plug-ins

The compound event plug-in is being tested for both the common cause module, utility module (i.e. add, multiply), and load-capacity. The scenarios include testing the utility module and load-capacity, testing the add and multiply functions in order to calculate the probability of the compound event. Then change sets are created to affect the compound event and the final probability. The results are verified to make sure the probability is correct. Also tested is the load-capacity plug-in. The values are input and the probability is calculated along with performing an uncertainty calculation. The input values are also modified using a change set and then a new probability along with uncertainty evaluation is performed and verified to be correct.

2.3.1.58 Test-58, Base Case Update

All models have fault tree results and accident sequences cut sets copied to the base case (this is still the case in SAPHIRE 8). This scenario is for fault tree cut sets copied to the base case for comparison to the current case using change sets.

2.3.1.59 Test-59, Calculation Types

The calculation types are tested. The “TRUE” calculation type is tested. The “TRUE, FALSE, and IGNORE” calculation types are tested. Fault trees are developed to verify the different calculation types are being changed in the change sets and the results are correct. The other calculation types (i.e., 3, 5, and 7) are also being checked in the simple database using change sets.

2.3.1.60 Test-60 Application of Change Sets

Change sets are used in numerous databases. Both class and single event change sets are developed and tested. The change sets test both probability changes and calculation type changes.

2.3.1.61 Test-61, Uncertainty Distributions

All of the uncertainty distribution types are tested.

2.3.1.62 Test-62, N of M Gates

The N/M gates are tested using the simple database (SIMPLE-FT) plant model. The N/M gate has multiple N/M gates feeding into it. The N/M gate is evaluated using all of the inputs and also with inputs affected by change sets.

2.3.1.63 Test-63, Sequence Stress Testing

Several scenarios test sequence stress (i.e., numerous sequences being generated). An event tree links over and over in order to test the ability to generate numerous sequences correctly.

2.3.1.64 Test-64, Calculations on the Common-Cause Plug-in

Use of the common-cause plug-ins is verified. Basic events are tested by using change sets. One set of the inputs is set TRUE. This requires SAPHIRE to re-calculate the Common Cause Failure (CCF) plug-in basic event for evaluation. The final probability is manually calculated and checked to the probability calculated for final verification.

2.3.1.65 Test-65, Event Transformations

Use the event transformations to ensure that the various model types of the basic event are represented in the cut sets. This would be a cut set level review. Note that this feature is still tested even though the option does not appear in SAPHIRE 8 and may be formally deprecated in future releases of Version 8.

2.3.1.66 Test-66, Wrong Results

This test verifies the output of results. The output from the test is compared to known incorrect results to verify that the comparison function worked correctly. A LOSP scenario is executed for comparison.

2.3.1.67 Test-67, Event and Condition Analysis-Initiating Event Assessment: Switchyard-Related Loss of Off-Site Power (LOOP) with other failures and conditions on the Oyster Creek 345 model

Scenarios exercise Event and Condition Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation,

quantification, and recovery of cut sets. One of the scenarios will test the T&M left in the model and the other will test with the T&M events removed.

2.3.1.68 Test 68, Event and Condition Analysis-Condition Event Assessment: Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model

Scenarios exercise Event and Condition Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for condition assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option. The test will need to verify that the CCF probability for a group of N was recalculated to be a CCF probability of group N-1 when a test and maintenance basic event is set to TRUE.

2.3.1.69 Test 69, Significance Determination Process -Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model

Scenarios exercise Significance Determination Process workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for Significance Determination Process assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option.

2.3.1.70 Test 70, General Analysis-Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model

Scenarios exercise General Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for Significance Determination Process assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option.

2.3.1.71 Test 71, 'N' Calculation type

When all projects are upgraded to version 8, all initiating events calculation types are changed from calculation type '1' to calculation type 'N'. Scenarios compare results in a project to prove that the 'N' calculation type upgrade has not changed results.

2.3.1.72 Test 72, RASP Common Cause Failure (CCF) validation

Scenarios compare results (in Standard Analysis) in a project to prove that the RASP-CCF 'R' calculation type upgrade works properly and provides expected results with both rolled-up and expanded output. These test results will need to be verified by an expert in the RASP CCF field. Make sure the flag set adjustment is validated.

2.3.1.73 Test 73, External Event Models - Solve Event Trees

Scenarios link external event model event trees, generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.

2.3.1.74 Test 74, Shutdown Models - Solve Event Trees

Scenarios link shutdown model event trees, generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.

2.3.1.75 Test 75, Workspace model independence

Scenarios ensure that databases move properly into the workspace and that workspace information remains independent from other workspaces and do not impact the base model.

2.3.1.76 Test 76, Repetition of critical calculations over N times

Scenarios ensure that cut set solving and recovery when done N times in a row calculate the same cut sets and quantification values. Do for Standard Analysis, Event and Condition Analysis, Significance Determination Process, and General Analysis.

2.3.1.77 Test 77, Significance Determination Process -LERF multiplier calculations

Scenarios ensure that Significance Determination Process -LERF multipliers are being used properly to calculate Screening LERF values.

2.3.1.78 Test 78, Accident Sequence Matrix – Solve Event Trees

Scenarios link event trees after an Accident Sequence Matrix has been loaded, generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.

2.3.1.79 Test 79, Multiple pass algorithm test (True and 1.0)

Scenarios set one or more basic events to 1.0 and validate the generated cut sets to ensure proper cut set creation (both in Standard Analysis and ECA). They also will set one or more basic events to TRUE and validate the generated cut sets to ensure proper cut set creation.

2.3.1.80 Test 80, Multiple pass algorithm test (False and Ignore)

Scenarios set one or more basic events to False and validate the generated cut sets to ensure proper cut set creation. They also will set one or more basic events to Ignore and validate the generated cut sets to ensure proper cut set creation.

2.3.1.81 Test 81, Min-Max test on Demo-EE model for Standard Analysis and General Analysis interfaces

Scenarios quantify all the Demo-EE sequences using the min/max or inclusion/exclusion method to ensure the validity of the frequencies. One scenario will test it for the Standard Analysis and another scenario will test it for a General Analysis.

2.3.1.82 Test 82, Single pass algorithm tests on Event and Condition Analysis and General Analysis

Scenarios exercise Event and Condition Analysis and General Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for condition assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the Event and Condition Analysis interface and another will test the General Analysis interface.

2.3.1.83 Test 83, Cross-referencing is validated

Scenarios will exercise the various cross referencing capabilities.

2.3.1.84 Test 84, Verify database recovery works

Scenarios will exercise the database recovery capabilities.

2.3.1.85 Test 85, Verify event tree/fault tree transfers function correctly (Manual tests)

Scenarios will exercise the transfer functions.

2.3.1.86 Test 86, Gather End States on a demo model with multiple phases

Scenarios will exercise the end state gathering on a demo model with multiple phases.

2.3.1.87 Test 87, Large Early Release Frequency (LERF) model functionality

Scenario opens an existing LERF model and exercises the Standard Analysis interface. The LERF model will be one of the models created in SAPHIRE 7 to calculate LERF results (Peach Bottom or Surry).

2.3.1.88 Test 88, Event Tree, Fault Tree Creation in a new project.

Scenarios builds in the Standard Analysis interface a demonstration sized model with 3 phases and two model types from scratch and save the new project.

2.3.1.89 Test 89, Menu Navigation

Scenario tests the Significance Determination Process and the Event and Condition Analysis interfaces. The buttons for moving back and forth between screens, canceling, and saving will be tested.

2.3.1.90 Test 90, Fault Tree View Expanded

Scenario verifies that in the Standard Analysis interface the fault tree feature “View Expand” is working.

2.3.1.91 Test 91, Workspace to Standard Analysis Interface Independence

Scenarios create and save new Significance Determination Process, Event and Condition Analysis, and General Analysis workspaces. Standard Analysis should never see any impact from workspace activity other than noting any saved workspaces in the workspace window. Test on Significance Determination Process, Event and Condition Analysis, General Analysis.

2.3.1.92 Test 92, Standard Analysis Interface to Workspace Independence

Scenarios run a change set in the Standard Analysis interface and visually verify that running this change set does not alter existing workspaces. Test on Significance Determination Process, Event and Condition Analysis, General Analysis workspaces.

2.3.1.93 Test 93, Workspace to Workspace Independence

Verify that addition of a new workspace or editing existing (e.g., logic changes) workspaces do not impact other workspaces. Tests should verify that changes made to Standard Analysis after the creation of a workspace should not reflect that change. Test on Significance Determination Process, Event and Condition Analysis, General Analysis workspaces.

2.3.1.94 Test 94, Project to Project Independence

Scenarios verify that opening up a project does not include anything from a previously opened project database. Examine database to ensure previous database (different from the one just opened) information is not present.

2.3.1.95 Test 95, Workspace to Workspace Independence

Scenarios verify that information from a previous case run in a workspace is not showing up in the next created case.

2.3.1.96 Test 96, Integrated Models

Scenarios verify that Demo-EE model produces expected results in the Standard Analysis, the Event and Condition Analysis, and the General Analysis interfaces.

2.3.1.97 Test 97, Event Tree Linking and Unlink in integrated models (External Events and Shutdown)

Scenarios verify that the event tree linking and unlinking functions work properly for selected integrated models with External Events and Shutdown information.

2.3.1.98 Test 98, Verification of cut set view path

Scenarios verify all cut set path features work. The initial test will be visually verified with reports produced. Subsequent tests will compare reports to verified reports.

2.3.1.99 Test 99, Verification Rule layering works

Scenarios verify linkage, post- processing, partitioning, and slice rule layering works.

2.3.1.100 Test 100, Verification Rule Nesting works

Scenarios verify linkage, post- processing, partitioning, and slice rule nesting works.

2.3.1.101 Test 101, All reports produce expected reports

Scenarios exercise the production of key reports available in all the workspaces and interfaces. Initially key reports will be produced and validated and then key future tests will compare freshly produced reports to the validated ones.

2.3.1.102 Test 102, Significance Determination Process Interface testing of basic event changes

Scenarios exercise Significance Determination Process basic event testing. Ensure that choosing “True” performs a CCF probability calculation correctly and choosing “True” automatically handles T&M basic events correctly.

2.3.1.103 Test 103, Significance Determination Process Interface testing of Figure III-D (Change in delta core damage frequency CDF as a function of duration) point estimate checks

Scenarios exercise Significance Determination Process workspace output. Figure III-D Change in delta CDF as a function of duration) is based upon a one hour value expanded to a full year outage. Test to determine that these point estimates are correct.

2.3.1.104 Test 104, Event and Condition Analysis uncertainty calculations

Scenarios exercise Event and Condition Analysis workspace uncertainty calculations and corresponding graph of the Importance = CCDP - CDP.

2.3.1.105 Test 105, SPAR-H worksheet calculations

The SPAR-H calculation type X will be check to ensure the correct probability is being produced. The basic events from a SPAR model that use this calculation type will be used for the test, and will include diagnosis, action, and dependency calculations.

3. CONCLUSIONS

Product quality is a key component of SAPHIRE. The SAPHIRE QA processes documented in the report provides the basis for setting quality objectives, progress, and the necessary framework for quality improvements. The QA plan will evolve as the SAPHIRE product is enhanced to provide the end user with solutions to their technical problems and cost-effectively meet user expectations. A majority of the changes within the SAPHIRE software occur because the end user has identified characteristics that provide "new potential," thus resulting in SAPHIRE evolving as each new feature is discovered and implemented. Therefore, the majority of software maintenance comes about not because of deficiencies in the code, but because it was modified to embrace improved methods for risk and reliability assessment or to take advantage of changes in software development practices.

SAPHIRE implements the key components needed to assure product quality. Management enables the software development team to apply a graded approach to effectively tailor activities, techniques, and methodologies to provide for:

- Configuration Management and Change Control
- Software defect reporting
- Software evolution and enhancement
- Corrective, preventive, and adaptive maintenance
- Deriving detailed requirements from the requirements and design direction obtained from contract documents.
- Development of test cases and scenarios and their implementation into an automated test suite used for comprehensive testing to assure that requirements are validated
- Recording and implementing lessons learned

These factors provide the necessary assurance that quality is "built-in" to the SAPHIRE software, not "tested in." Quality must be planned, designed, implemented and verified before it can be validated through the testing process. SAPHIRE will continue to be evaluated for quality as it evolves. As such, this quality plan will also evolve as the needs and goals of the user and customer evolve to ensure the dimensions of quality are established and assessed.

4. REFERENCES

Bolander, T. W. et al., (1994) *Verification and Validation of the SAPHIRE Version 4.0 PRA Software Package*, NUREG/CR-6145, February.

Jones, J. L. et al., (1995) *Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 5.0 Verification and Validation (V&V) Manual*, NUREG/CR-6116, February.

Smith, C.L. et al, (2000) *Testing, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0*, NUREG-CR/6688, September.

Smith, C. L., R. Nims, K. J. Kvarfordt, C. Wharton, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 6 Quality Assurance Manual*, NUREG/CR-6952, August 2008.

US NRC, (1993) *Software Quality Assurance Program and Guidelines*, NUREG/BR-0167, February.

Appendix A

SAPHIRE Salient Features List

Appendix A

SAPHIRE Salient Features List

In order to provide additional context to the complexity of a modern analysis code such as SAPHIRE (and its associated implications on testing) included is the list of salient features found in the software in Table A-1.

Table A-1 SAPHIRE Salient Features as a Function of the Version Number

Item	Description of Feature	Version 7.x	Version 8.x
A	Cut Set Sequence Generation		
A.1	Rule-based Fault Tree Linking	X	X
A.2	Linking of Small Tree Events	X	X
A.3	Linking of Large Tree Events	X	X
A.4	Sequence Capacity	2 million	2 million
B	Cut Set Generation		
B.1	Fault Trees	X	X
B.2	Event Trees	X	X
C	Cut Set Gathering		
C.1	Sequence End States	X	X
C.2	Sequence End State Cut Sets	X	X
D	Cut Set Partitioning		
D.1	End State Definition by rules	X	X
E	Cut Set Slice		
E.1	By Event	X	X
E.2	By Probability	X	X

Item	Description of Feature	Version 7.x	Version 8.x
E.3	By Rules	X	X
E.4	Multiple sequential slices		X
E.5	Enhanced cut set slice viewer		X
F	Cut Set Postprocessing (Recovery)		
F.1	Event Trees	X	X
F.2	Fault Trees	X	X
F.3	Ability to layer rule application		X
G	Change Sets (Selected subset of Basic Events for temporary analysis)		
G.1	Single event selection	X	X
G.2	Multiple event selection	X	X
G.3	Group event selection	X	X
H	Flag Sets (Selected subset of Basic Events with logic changes only)		
H.1	Cut Set with Static Flag Sets	X	X
H.2	Cut Set with Dynamic Flag Sets (linkage rules)	X	X
H.3	Applicable to Fault Trees	X	X
H.4	Applicable to Sequences	X	X
H.5	Applicable to Fault Trees within Sequences	X	X
H.6	Flag sets can affect common-cause events		X
I	Cut Set Quantification Methods		
I.1	Minimal cut set upper-bound	X	X
I.2	Min-Max	X	X

Item	Description of Feature	Version 7.x	Version 8.x
I.3	Rare Event	X	X
I.4	Split Fraction (sequences only)	X	X
I.5	Binary Decision Diagram (fault trees)		X
J	Cut Set Analysis		
J.1	Cut Set Verification – cut sets solved, gathered, with truncation by size or probability, auto recovery	X	X
J.2	Cut Set path tracing	X	X
J.3	Cut Set comparison	X	
J.4	Fault Tree	X	X
J.5	Event Trees / Sequences	X	X
J.6	End States	X	X
K	Basic Event Management		
K.1	Basic Events – Generation	X	X
K.2	Basic Event – Templates	X	X
K.3	Multiple basic event editing at the same time		X
L	Basic-Event Calculations		
L.1	Compound Events Common-cause plug-in modules		
L.1.1	Common-cause alpha-factor module	X	X
L.1.2	Common-cause beta-factor module	X	X
L.1.3	Common-cause capacity load module	X	X
L.1.4	Common-cause multiple Greek letter module	X	X
L.1.5	Common-cause multiple group module	X	X
L.1.6	Common-cause alpha-staggered module	X	X

Item	Description of Feature	Version 7.x	Version 8.x
L.1.7	Common-cause RASP expanded module		X
L.1.8	Loss-of-offsite power module	X	X
L.1.9	Time series module	X	X
L.1.10	General calculation module	X	X
L.2	Failure Probability on Demand	X	X
L.3	Failure Probability to Run	X	X
L.4	Value input (for any value)	X	X
L.5	Failure Probability to Run w/ repair	X	X
L.6	Failure Probability to Run	X	X
L.7	House Event True (Prob = 1.0), i.e. failed	X	X
L.8	House Event False (Prob = 0.0), i.e. success	X	X
L.9	House Event Ignore	X	X
L.10	Compound Event	X	X
L.11	Human Factor Event	X	X
L.12	Fault tree Min Cut Upper Bound Value	X	X
L.13	End State Min Cut Upper Bound Value	X	X
L.14	Ground Acceleration Value	X	X
L.15	Hazard Curve	X	X
M	Importance Measures		
M.1	Fussell-Vesely Importance Measure	X	X
M.2	Birnbaum Importance Measure	X	X
M.3	Risk increase ratio importance measure	X	X
M.4	Risk reduction ratio importance measure	X	X

Item	Description of Feature	Version 7.x	Version 8.x
M.5	Risk increase interval importance measure	X	X
M.6	Risk reduction interval importance measure	X	X
M.7	Group importance measure	X	X
M.8	Uncertainty determination on Importance Measures	X	X
N	Model Creation		
N.1	Fire and flooding capability	X	X
N.2	Fault Tree text editor	X	
N.3	Drag-and-drop Fault Tree graphical editor		X
N.4	Event Tree text editor	X	
N.5	Event Tree graphical editor	X	X
N.6	Drag-and-drop Event Tree editor		X
N.7	Basic Load / Extract Data Models	X	X
N.7.1	Extract All	X	X
N.7.2	Load All	X	X
N.7.3	Extract All File types	X	X
N.7.4	Load All / Group	X	X
N.7.5	Fault Tree Logic	X	X
N.7.6	Designate output folder location	X	X
N.8	Graphical Export to Windows metafiles	X	X
N.9	Graphical Export to JPEG		X
N.11	Database Recovery	X	X
N.12	Database MAR-D Load and Extract	X	X
N.12.1	Event Tree MAR-D	X	X

Item	Description of Feature	Version 7.x	Version 8.x
N.12.2	Fault Tree MAR-D	X	X
N.12.3	Basic Event MAR-D	X	X
N.13	Macro manager		X
N.14	Alternate names and descriptions for all database objects (for multilingual use)	X	X
N.15	Model Version Upgrade (backward compatible)	X	X
N.16	Integrate two projects into single project		X
N.17	Creation of external events model via Accident Sequence Matrix file		X
N.18	Phase-aware basic events and event trees		X
N.19	User definable grouping of event trees		X
O	Model Creation Logic Gate Types (Maximum inputs 256 unless otherwise specified)		
O.1	AND	X	X
O.2	OR	X	X
O.3	N of M (Max N=98 , Max M=99)	X	X
O.4	NAND (Not AND)	X	X
O.5	NOR (Not OR)	X	X
O.6	Transfer Gate	X	X
O.7	Inhibit gate	X	
P	Uncertainty Calculations (Monte Carlo and Latin Hyper Cube Sampling)		
P.1	None (or Point Value only)	X	X
P.2	Normal Distribution	X	X

Item	Description of Feature	Version 7.x	Version 8.x
P.3	Lognormal Distribution	X	X
P.4	Beta Distribution	X	X
P.5	Chi Squared Distribution	X	X
P.6	Exponential Distribution	X	X
P.7	Uniform Distribution	X	X
P.8	Constrained non-informative Distribution	X	X
P.9	Gamma Distribution	X	X
P.10	Maximum Entropy Distribution	X	X
P.11	Dirichlet Distribution	X	X
P.12	Seismic Log Normal analysis	X	X
P.13	Histogram Distribution	X	X
P.14	Triangular Distribution	X	X
Q	Uncertainty Calculations (Parameter Settings)		
Q.1	Seed	X	X
Q.2	Sample Size	X	X
Q.3	Built in cumulative and density plots		X
R	General Support Features		
R.1	Sensitivity Wizard	X	X
R.2	Importance Measures Wizard	X	X
R.3	Embedded Macro capability	X	X
R.4	Editing User Information	X	X
R.5	Page numbering control on graphic format	X	X
R.6	Conversion from alpha to graphic format	X	n/a

Item	Description of Feature	Version 7.x	Version 8.x
R.7	On-line Context Sensitive help	X	X
R.8	User customizable icons calling analysis icons		X
R.9	Multiple editing and reporting windows open at the same time		X
R.10	Project Check		X
R.11	Project-wide search ability		X
R.12	Bookmarking of object lists		X
R.13	Drag-and-drop flag and change set creation		X
R.14	Support for opening and creating compressed (zip) project files		X
S	General Support Features (Report Generation)		
S.1	Project Reports	X	X
S.2	Attributes	X	X
S.3	Basic Event	X	X
S.4	Fault Tree	X	X
S.5	Event Tree	X	X
S.6	End State	X	X
S.7	Sequence	X	X
S.8	Change Set	X	X
S.9	Flag Set	X	X
S.10	Gate	X	X
S.11	Histogram	X	X
S.12	Slice	X	X
S.13	User Info	X	X

Item	Description of Feature	Version 7.x	Version 8.x
S.14	Cross Reference Reports	X	X
T	Report Format Types		
T.1	ASCII	X	X
T.2	RTF	X	X
T.3	HTML	X	X
T.4	Acrobat™ PDF		X
U	General Analysis Types		
U.2	Initiating Event Analysis	X	X
U.3	Condition Assessment Analysis	X	X
U.4	Accident Sequence Precursor	X	X
U.4.2	General analysis types	X	X
U.4.3	Load-capacity calculation module	X	X
U.5	Significance Determination Process		X
V	Application Program Interface		
V.1	Microsoft Visual Basic™ and VBA interface	X	X
V.2	Microsoft Visual C\C++™ interface	X	X
V.3	Borland Delphi™	X	X
W	Operating Systems		
W.1	Windows XP™	X	X
W.2	Windows Vista™	X	X
W.3	Window 7™	X	X

APPENDIX B

SAPHIRE QA Process Checklist and Change Forms

APPENDIX B – SAPHIRE QA Process Checklist and Change Forms

The project manager provides monthly reports, draft reports, and final TV&V report to the SAPHIRE sponsor of completed and pending maintenance tasks.

OK		Comments:
Discrepancy		
N/A		

The development team obtains and retains change request information.

OK		Comments:
Discrepancy		
N/A		

The development team obtains and reviews documented lessons learned from previous development efforts.

OK		Comments:
Discrepancy		
N/A		

Requirements derived from NRC requirements documents are verified and validated for implementation into automated test scripts.

OK		Comments:
Discrepancy		
N/A		

NRC requirements documents provide the requirements needed for software enhancements. Questions regarding any requirement specified by these forms are obtained from the appropriate NRC representative and the clarification of any requirement is documented and placed under configuration control.

OK		Comments:
Discrepancy		
N/A		

Detailed requirements are derived from the higher-level requirements provided within the NRC forms.

OK		Comments:
Discrepancy		
N/A		

Detailed requirements and the code, test scripts, and test results are validated to ensure that all requirements were implemented and tested.

OK		Comments:
Discrepancy		
N/A		

The designated QA inspector reviews completed and pending tasks for compliance to requested enhancements or other maintenance activities, such as bug fixes.

OK		Comments:
Discrepancy		
N/A		

A TV&V document is developed and includes implemented requirements, new features, bug fixes and test results.

OK		Comments:
Discrepancy		
N/A		

Prior to an official release, software is processed through a series of automated test scripts.

OK		Comments:
Discrepancy		
N/A		

Test scripts simulate typical user input.

OK		Comments:
Discrepancy		
N/A		

Models suitable for testing one or more critical functions consist of actual PRA models.

OK		Comments:
Discrepancy		
N/A		

Test results are saved and compared against expected results.

OK		Comments:
Discrepancy		
N/A		

User documentation is updated upon completion of each new release.

OK		Comments:
Discrepancy		
N/A		

Software releases are bundled into a software installation package for use in set-up.

OK		Comments:
Discrepancy		
N/A		

Software releases include list of bug fixes, new features, and historical information.

OK		Comments:
Discrepancy		
N/A		

Only authorized changes are made to the software release.

OK		Comments:
Discrepancy		
N/A		

Software and supporting documentation is baselined and placed under configuration control.

OK		Comments:
Discrepancy		

N/A		

The software librarian (or designee) places all baselined data, including builds generated during development, software fixes and enhancements, and software releases under configuration control via the configuration management database.

OK		Comments:
Discrepancy		
N/A		

The configuration management database precludes users from simultaneously accessing the same information.

OK		Comments:
Discrepancy		
N/A		

Prior to check in of information obtain from the configuration library database, users provide an explanation of any changes made.

OK		Comments:
Discrepancy		
N/A		

Step-by-step instructions obtained from end users reporting bugs/defects are used to reproduce the process that generated the bug. This information is placed under configuration control.

OK		Comments:
Discrepancy		
N/A		

Bugs are categorized by severity.

OK		Comments:
Discrepancy		
N/A		

Change requests and bug fixes are placed under configuration control.

OK		Comments:
Discrepancy		
N/A		

Version control software tracks changes by author and time.

OK		Comments:
Discrepancy		
N/A		

The automated software process generates a summary report, detail report, test identification number, description, and pass/fail indicator.

OK		Comments:
Discrepancy		
N/A		

Generation of new test scripts include obtaining information solicited/received from experienced users and are examined to determine importance and testability.

OK		Comments:
Discrepancy		

N/A		

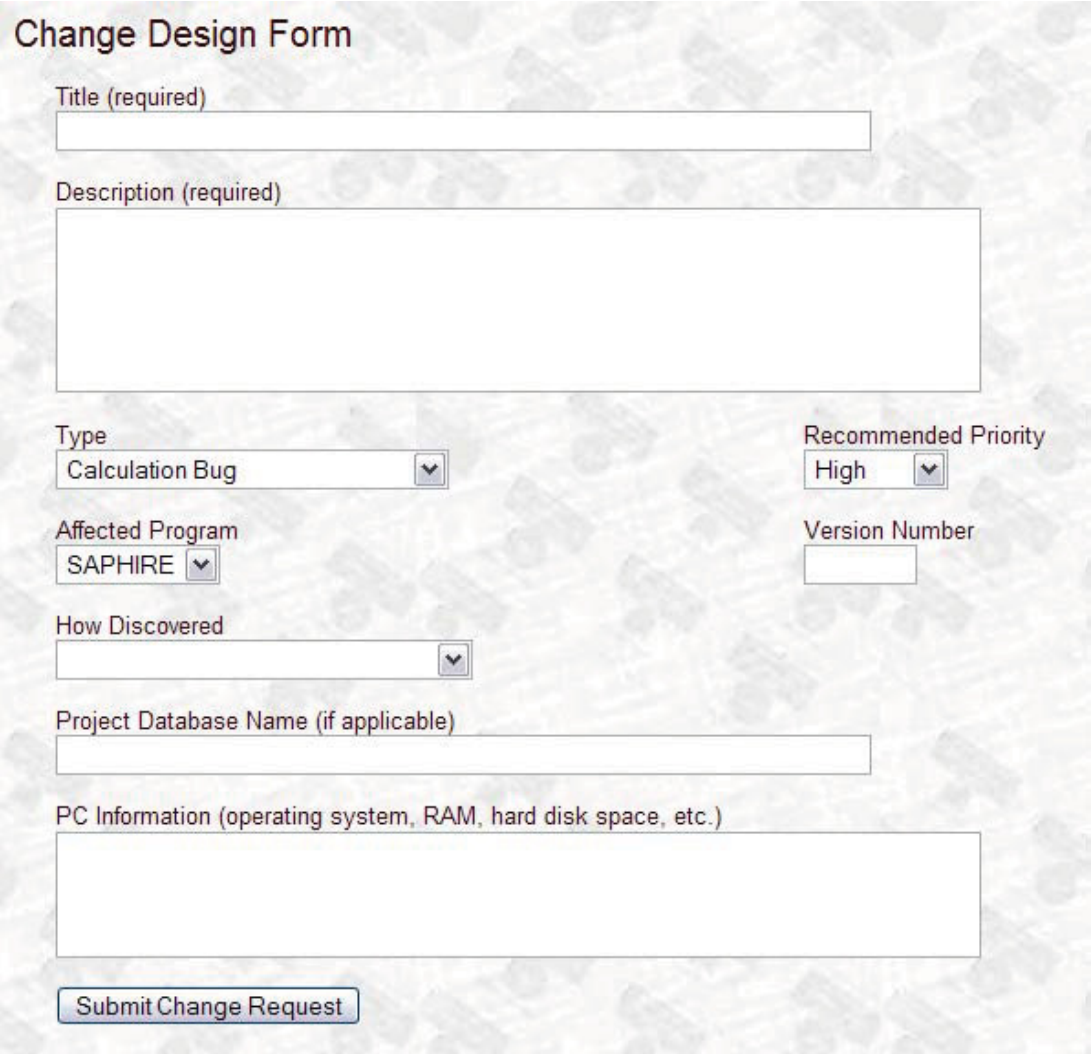
Test scripts are reviewed to ensure that requirements are tested adequately, completely, and correctly.

OK		Comments:
Discrepancy		
N/A		

When a bug is reported, the user should gather and record the relevant information about the bug on the change request form (see below). General information should include bug reporter contact information and program version information.

System environment information such as operating system and available memory and disk information should be collected as well, when it appears this information may be a factor into the error.

The problem should be described in sufficient detail as to allow the programmer to reproduce the error. The programmer may request that the bug reporter isolate the problem as much as possible. When necessary, a database should be provided with step by step instructions on how to reproduce the bug.



As the change information is collected, the problem should be categorized as a major bug, minor bug, improvement, or new feature:

- A major bug is defined as an error that stops the user from completing a task and/or adversely affects the core calculation ability of SAPHIRE.

- A minor bug is defined as an error for which a work around is available, or something that affects less essential areas of SAPHIRE, such as a slight user interface malfunction.

- The improvement category is defined as a change that will represent added convenient to the user. For this category, the change is not significant enough to be considered a new feature. Examples of improvements are minor report enhancements, and replacing or adding smoother user interface options.

- A new feature is defined as a significant additional capability to be added. The scope of a new feature is greater than that of an improvement to an existing feature.

Examples of new features include new calculation or uncertainty types, new wizards, and new plug-ins.

The priority of a change will generally correlate with the category of the change. Major bugs are generally the highest priority. Minor bugs and suggested improvements are medium to low priority, depending on the pervasiveness of the problem. Customers and project management together prioritize new features.

APPENDIX C

SAPHIRE/GEM Test Suite Summary Report

APPENDIX C – SAPHIRE/GEM Test Suite Summary Report

The tests that are in the SAPHIRE TV&V automated test suite are listed in Table C-1. The status of each test, on a pass/fail basis, is reported in this table. Problems associated with failures, if any, are investigated and corrected prior to a release of the software.

Table C-1. SAPHIRE TV&V Automated Tests

Test #	Test Name	Description	Pass or Fail?
Test-01	Solve Fault Trees	Scenarios generate basic event data (with no change sets) solve (with cut set probability cutoff) and quantify fault tree minimal cut sets and recovery rules. The base case min cut upper bound, alternate case min cut upper bound, and cut set totals are verified for each fault tree.	Pass
Test-02	Core Damage Frequency	Scenarios generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The alternate case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.	Pass
Test-03	Events and Condition Assessment: Auxiliary Feed Water out of service for 72 hours	Scenarios exercise all aspects of operational event analysis including removal of equipment from service and automated processing of all steps. These steps include basic event generation with change sets; and generation, quantification, and recovery of cut sets. The number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified.	Pass
Test-04	Events and Condition Assessment: Emergency Diesel Generator out of service for three months	Scenarios exercise all aspects of operational event analysis, including removal of equipment from service and automated processing of all steps. These steps include basic event generation with change sets, and generation, quantification, and recovery of cut sets. The number of sequences, total CCDP, total core damage probability (CDP), total importance, and CCDP, CDP, and importance for each sequence are verified.	Pass
Test-05	Initiating Event Assessment: Transient with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-06	Initiating Event Assessment: Small Loss of Coolant Accident (SLOCA) with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets	Pass
Test-07	Initiating Event Assessment: Steam Generator Tube Rupture with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets	Pass
Test-08	Initiating Event Assessment: Grid-Related Loss of Off-Site Power (LOOP) with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.	Pass (old LOOP tests are deprecated)
Test-09	Initiating Event Assessment: Plant-Centered LOOP with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.	Pass (old LOOP tests are deprecated)
Test-10	Initiating Event Assessment: Severe Weather LOOP with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.	Pass (old LOOP tests are deprecated)
Test-11	Initiating Event Assessment: Extreme Severe Weather LOOP with no other failures	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets.	Pass (old LOOP tests are deprecated)
Test-12	Initiating Event Assessment: Transient with AFW Failed	Scenarios exercise the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation,	Pass

Test #	Test Name	Description	Pass or Fail?
		quantification, and recovery of cut sets.	
Test-13	Dominant Sequence Frequencies and Core Damage Frequency Uncertainty	This scenario continues the tracking with an automated test script. Cut sets generated with cut set probability cutoff and cut set size cutoff. Recovery rules are applied without cutoff. Cut set update performed with no truncation. Project level Monte Carlo uncertainty performed on results using 5000 samples.	Pass
Test-14	Fault Tree Uncertainty: Monte Carlo Method/Log Normal Distribution	This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the log normal distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and error factors. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples (simulated values) and a random number seed of 4,321 for each test.	Pass
Test-15	Fault Tree Uncertainty: Monte Carlo Method/Normal Distribution	This scenario consists of variations that test uncertainty using the Monte Carlo simulation technique for the normal distribution type. Two fault trees are used that consist of an OR gate with a single basic event as its input, with differing basic event nominal probabilities and standard deviation values. Fault tree combinations of five sample sizes and two seed values are used for a total of ten tests for each tree. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.	Pass
Test-16	Fault Tree Uncertainty: Monte Carlo Method/Beta Distribution	This scenario consists of ten variations that test uncertainty using the Monte Carlo simulation technique for the beta distribution type. The ten variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-17	Fault Tree Uncertainty: Monte Carlo Method/Chi Squared Distribution	This scenario consists of twelve variations that test uncertainty using the Monte Carlo simulation technique for the chi-square distribution type. For ten of the variations, ten fault trees are used that consists of an OR gate with a single basic event as its input. Each basic event has a different nominal probability and uncertainty value (degrees of freedom). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test. For the other variations two fault trees are used that consist of an OR gate with a single basic event as its input with differing basic event nominal probabilities and uncertainty values. For each of these fault trees, four different sample sizes and seed of 4,321 are used. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.	Pass
Test-18	Fault Tree Uncertainty: Monte Carlo Method/Exponential Distribution	This scenario consists of eight variations that test uncertainty using the Monte Carlo simulation technique for the exponential distribution type. The eight variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-19	Fault Tree Uncertainty: Monte Carlo Method/Uniform Distribution	This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the uniform distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-20	Fault Tree Uncertainty: Monte Carlo Method/Gamma Distribution	This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the gamma distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values (r). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-21	Fault Tree Uncertainty: Monte Carlo Method/Maximum Entropy Distribution	This scenario consists of seven variations that test uncertainty using the Monte Carlo simulation technique for the maximum entropy distribution type. The seven variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end and lower end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-22	Sequence Uncertainty: Monte Carlo Method / Dirichlet Distribution	This test scenario consists of four variations that test uncertainty analyses using the Monte Carlo simulation technique for the Dirichlet distribution type. The first three variations each use a three-branch event tree with differing failure probabilities and parameter values. The fourth variation uses a 121-branch event tree. Change sets are used to correlate the basic events. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.	Pass
Test-23	Fault Tree Uncertainty: Monte Carlo Method/Seismic Distribution	This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the seismic distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event median failure acceleration, screening G-level, Beta-R and Beta-U values. Uncertainty analysis is performed using the Seismic analysis type. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 samples and a seed of 4,321 for each test.	Pass
Test-24	Fault Tree and Sequence Uncertainty: Monte Carlo Method/Constrained Noninformative Distribution	This scenario consists of five variations that test uncertainty using the Monte Carlo simulation techniques for the Constrained Noninformative distribution type. The three variations involving fault trees use fault trees that consists of an OR gate with a single basic event as its input with differing basic event nominal probabilities. The two variations involving sequences use event trees with differing initiating event nominal frequencies. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 simulated values for each test.	Pass
Test-25	Fault Tree Uncertainty: Latin Hypercube Method/Log Normal Distribution	This scenario consists of six variations that test uncertainty using the Latin Hypercube simulation technique for the log normal distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and error factors. The 5th percentile, 50th percentile, 95th	Pass

Test #	Test Name	Description	Pass or Fail?
		percentile, and standard deviation results are verified based on 5,000 samples (simulated values) and a random number seed of 4,321 for each test.	
Test-26	Fault Tree Uncertainty: Latin Hypercube Method/Normal Distribution	This scenario consists of variations that test uncertainty using the Latin Hypercube simulation technique for the normal distribution type. Two fault trees are used that consist of an OR gate with a single basic event as its input, with differing basic event nominal probabilities and standard deviation values. Fault tree combinations of five sample sizes and two seed values are used for a total of ten tests for each tree. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.	Pass
Test-27	Fault Tree Uncertainty: Latin Hypercube Method/Beta Distribution	This scenario consists of ten variations that test uncertainty using the Monte Carlo simulation technique for the beta distribution type. The ten variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-28	Fault Tree Uncertainty: Latin Hypercube Method/Chi Squared Distribution	This scenario consists of twelve variations that test uncertainty using the Monte Carlo simulation technique for the chi-square distribution type. For ten of the variations, ten fault trees are used that consists of an OR gate with a single basic event as its input. Each basic event has a different nominal probability and uncertainty value (degrees of freedom). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test. For the other variations two fault trees are used that consist of an OR gate with a single basic event as its input with differing basic event nominal probabilities and uncertainty values. For each of these fault trees, four different sample sizes and seed of 4,321 are used. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified.	Pass
Test-29	Fault Tree Uncertainty: Latin Hypercube Method/Exponential Distribution	This scenario consists of eight variations that test uncertainty using the Monte Carlo simulation technique for the exponential distribution type. The eight variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities. The 5th percentile, 50th	Pass

Test #	Test Name	Description	Pass or Fail?
		percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	
Test-30	Fault Tree Uncertainty: Latin Hypercube Method/Uniform Distribution	This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the uniform distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-31	Fault Tree Uncertainty: Latin Hypercube Method/Gamma Distribution	This scenario consists of six variations that test uncertainty using the Monte Carlo simulation technique for the gamma distribution type. The six variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and uncertainty values (r). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-32	Sequence Uncertainty: Latin Hypercube Method/Maximum Entropy Distribution	This scenario consists of seven variations that test uncertainty using the Monte Carlo simulation technique for the maximum entropy distribution type. The seven variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and upper end and lower end uncertainty values. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-33	Sequence Uncertainty: Latin Hypercube Method/Dirichlet Distribution	This test scenario consists of four variations that test uncertainty analyses using the Monte Carlo simulation technique for the Dirichlet distribution type. The first three variations each use a three-branch event tree with differing failure probabilities and parameter values. The fourth variation uses a 121-branch event tree. Change sets are used to correlate the basic events. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified. Since this distribution type was not available in version 5, version 6 results have been inspected for acceptance and are used for comparison against subsequent incremental releases.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-34	Fault Tree Uncertainty: Latin Hypercube Method/Seismic Distribution	This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the seismic distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event median failure acceleration, screening G-level, Beta-R and Beta-U values. Uncertainty analysis is performed using the Seismic analysis type. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 samples and a seed of 4,321 for each test.	Pass
Test-35	Fault Tree and Sequence Uncertainty: Latin Hypercube Method/ Constrained Noninformative Distribution	This scenario consists of five variations that test uncertainty using the Monte Carlo simulation techniques for the Constrained Noninformative distribution type. The three variations involving fault trees use fault trees that consists of an OR gate with a single basic event as its input with differing basic event nominal probabilities. The two variations involving sequences use event trees with differing initiating event nominal frequencies. The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 10,000 simulated values for each test.	Pass
Test-36	Fault Tree Uncertainty: Monte Carlo Method/Histogram Distribution	This scenario consists of four variations that test uncertainty using the Monte Carlo simulation technique for the histogram distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and histograms (of percentage, area, and range types). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-37	Fault Tree Uncertainty: Latin Hypercube Method/Histogram Distribution	This scenario consists of four variations that test uncertainty using the Latin Hypercube simulation technique for the histogram distribution type. The four variations use fault trees that consists of an OR gate with a single basic event as its input. Each variation uses differing basic event nominal probabilities and histograms (of percentage, area, and range types). The 5th percentile, 50th percentile, 95th percentile, and standard deviation results are verified based on 5,000 samples and a seed of 4,321 for each test.	Pass
Test-38	Gathering of End States	This scenario generates basic event data (with no change sets) and gathers the end states (without cut set probability cutoff, by sequence end state). The alternate case min-cut upper bound and the number of cut sets are verified for each end state.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-39	End State Uncertainty: Monte Carlo Method	These scenarios perform multiple event sampling on all sequences that belong to a particular end state (single uncertainty), as well as the collection of all end states (group uncertainty). The mean, 5th percentile, median, 95th percentile, and standard deviation results are verified based on 3,000 simulated values for each test.	Pass
Test-40	End State Uncertainty: Latin Hypercube Method	These scenarios perform multiple event sampling on all sequences that belong to a particular end state (single uncertainty), as well as the collection of all end states (group uncertainty) . The mean, 5th percentile, median, 95th percentile, and standard deviation results are verified based on 3,000 simulated values for each test.	Pass
Test-41	Cut Set Verification	This test case consists of scenarios that compare cut sets from selected fault trees, sequences, and end states. The cut set frequency, percent contribution to the total, and basic events in the cut set are verified. Cut sets are solved and /or gathered with truncation, auto-recovered, and updated. Sequences and fault trees are solved with and without their default flag sets. Also, fault tree editing is briefly tested. This is done by opening the alphanumeric logic editor, saving and converting logic to graphics, then pulling up the graphical editor and saving the graphics. This test does not test specific editing features but it does verify that the original logic is correctly loaded and saved. Failure of the logic to be preserved correctly would be detected with incorrect cut set results	Pass
Test-42	Link Small Event Tree	This scenario uses the Surry Large Early Release Frequency (LERF) Level 2/3 model (S_LERF) to link event trees using the small event tree methodology. Prior to link, each event tree is loaded into the graphical editor and saved to ensure that the correct logic is preserved. The sequences are then solved with cutoff. The alternate case min cut upper bound and number of cut sets is verified for each Level 1 sequence.	Pass
Test-43	Partition Sequence Cut Sets	This scenario applies event tree partition rules to the sequences generated in scenario reference number Test-42. These partition rules assign Plant Damage States (PDSs) to all sequences with cut sets. These end states are then gathered by cut set partition. The alternate case min cut upper bound and number of cut sets is verified for each PDS.	Pass
Test-44	Link Large Event Tree	This scenario uses the results from scenario reference number Test-43. The PDS event trees created by the partition rules are linked using the large event tree methodology and create sequence logic cut sets. The LERF end states are then gathered by sequence end state and re-quantified using the Rare Event approximation. The alternate case min-cut upper bound	Pass

Test #	Test Name	Description	Pass or Fail?
		and number of cut sets are verified for each LERF end state.	
Test-45	Fault Tree Importance Measures	This test case consists of scenarios that test importance measures calculations with fault trees for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.	Pass
Test-46	Sequence Importance Measures	This test case consists of scenarios that test Sequence importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.	Pass
Test-47	Sequence Group Importance Measures	This test case consists of scenarios that test Sequence Group importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.	Pass
Test-48	End State Importance Measures	This test case consists of scenarios that test End State importance measure calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.	Pass
Test-49	End State Group Importance	This test case consists of scenarios that test End State Group importance measures calculations for each of the importance measures: ratio, difference, and uncertainty. For each event, the name, number of occurrences, probability, Fussell-Vesely (or Birnbaum or uncertainty importance), risk reduction ratio (or difference), risk increase ratio (or difference) results are verified.	Pass
Test-50	Change Set Processing: Single	This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, single basic event changes are made in a change set. The change set is then marked and the basic event data is generated. An affected sequence is then selected and cut set results are verified.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-51	Change Set Processing: Class	This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, class basic event changes are made in a change set. The change set is then marked and the basic event data is generated. An affected sequence is then selected and cut set results are verified.	Pass
Test-52	Change Set Processing: Marked Order	This test case consists of scenarios that test the effects of basic event changes, via change sets, on sequence cut set results. In these scenarios, the change sets created in Test-50 and Test-51 are used. Multiple change sets are marked and the basic event data is generated. An affected sequence is then selected and cut set results are validated. This test verifies that the changed basic events are processed correctly based on the marked order of the change sets.	Pass
Test-53	Extract, Delete, Load, Solve: Fault Trees and Basic Events	This test consists of scenarios that exercise utility functions associated with the database for loading plant models, end state data or other information to be analyzed with the tool set.	Pass
Test-54	Fault Tree Utility Functions: Auto page, Solve, Save Cut Sets to End States	This scenario provides several utilities maintain fault trees. These tests verify that the use of these features does not introduce errors into the database. The auto-page scenario breaks up a large fault tree into manageable smaller fault trees with transfer information. An auto-page is performed on a large fault tree, and then the modified tree is solved to verify the cut set results are not altered with the paging operation. Another scenario copies a fault tree cut sets to an end state, and then verifies that the cut sets in the end state match the cut sets in the fault tree.	Pass
Test-55	Event Tree Linkage (including rules)	This scenario tests event tree linking rules using several different databases. The databases are the Surry LERF model, Wolf Creek Revision 302, and Peach Bottom Revision 302. The Surry LERF model links the Level 1 event tree sequences together prior to solving the accident sequences, then performs an end state gather. The end states then become Level 2 event trees, which are linked together using the large event tree method. These Level 2 sequences are then gathered into the final end states for LERF, NO-LERF, etc. The Wolf Creek and Peach Bottom models have no accident sequences at the beginning. The test has the sequences being generated using dynamic flag sets for the accident sequences, and then evaluates the sequences. The sequences are evaluated using the developed dynamic flag sets and then with no flag sets.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-56	End State Gathering	Scenario tests the end state gathering process using the Surry LERF model and the Beaver Valley NUREG 1150 model. Both models have the sequences gathered into end states. The Surry LERF model uses partition rules, while the Beaver Valley model uses the end state name.	Pass
Test-57	Compound Event Plug-ins	This scenario tests compound event plug-in for the common cause module, utility module (i.e. add, multiply), and load-capacity. The scenarios include testing the utility module and load-capacity, testing the add and multiply functions in order to calculate the probability of the compound event. Then change sets are created to affect the compound event and the final probability. The results are verified to make sure the probability is correct. Also tested is the load-capacity plug-in. The values are input and the probability is calculated along with performing an uncertainty calculation. The input values are also modified using a change set and then a new probability along with uncertainty evaluation is performed and verified to be correct.	Pass (based upon manual check of uncertainty results)
Test-58	Base Case Update	This scenario tests models that have fault tree results and accident sequences cut sets copied to the base case (this is still the case in SAPHIRE 8). This scenario is for fault tree cut sets copied to the base case for comparison to the current case using change sets.	Pass
Test-59	Calculation Types	The calculation types are tested. The "TRUE" calculation type is tested. The "TRUE, FALSE, and IGNORE" calculation types are tested. Fault trees are developed to verify the different calculation types are being changed in the change sets and the results are correct. The other calculation types (i.e., 3, 5, and 7) are also being checked in the simple database using change sets.	Pass
Test-60	Application of Change Sets	Change sets are used in numerous databases. Both class and single event change sets are developed and tested. The change sets test both probability changes and calculation type changes.	Pass
Test-61	Uncertainty Distributions	Various uncertainty distribution types are tested.	Pass
Test-62	N of M Gates	N/M gates are tested using the simple database (SIMPLE-FT) plant model. The N/M gate has multiple N/M gates feeding into it. The N/M gate is evaluated using all of the inputs and also with inputs affected by change sets.	Pass
Test-63	Sequence Stress Testing	These scenarios test sequence stress (i.e., numerous sequences being generated). An event tree links over and over in order to test the ability to generate numerous sequences correctly.	Pass

Test #	Test Name	Description	Pass or Fail?
Test-64	Calculations on the Common Cause Plug in	The function of the common-cause plug-ins is verified. Basic events are tested by using change sets. One set of the inputs is set TRUE. This requires SAPHIRE to re-calculate the Common Cause Failure (CCF) plug-in basic event for evaluation. The final probability is manually calculated and checked to the probability calculated for final verification.	Pass
Test-65	Event Transformations	Tests the event transformations to ensure that the various model types of the basic event are represented in the cut sets. This would be a cut set level review. Note that this feature is still tested even though the option does not appear in SAPHIRE 8 and may be formally deprecated in future releases of Version 8.	Deprecated
Test-66	Wrong Results	This test verifies the output of results. The output from the test is compared to known incorrect results to verify that the comparison function worked correctly. A LOSP scenario is executed for comparison.	Pass (test for "false positive")
Test-67	Event and Condition Analysis-Initiating Event Assessment: Switchyard-Related Loss of Off-Site Power (LOOP) with other failures and conditions on the Oyster Creek 345 model	This test exercise Event and Condition Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for initiating event assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the T&M left in the model and the other will test with the T&M events removed.	Pass
Test 68	Event and Condition Analysis-Condition Event Assessment: Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model	This test exercises Event and Condition Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for condition assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option. The test will need to verify that the CCF probability for a group of N was recalculated to be a CCF probability of group N-1 when a test and maintenance basic event is set to TRUE.	Pass

Test #	Test Name	Description	Pass or Fail?
Test 69	Significance Determination Process -Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model	This test exercises Significance Determination Process workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for Significance Determination Process assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option.	Pass
Test 70	General Analysis-Blue Max SBO Diesel out for four days on the Susquehanna Unit 1 and 2 model	This test exercises General Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for Significance Determination Process assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the single pass option and another will test the multiple pass option.	Pass
Test 71	'N' Calculation type	This test ensures that all projects are upgraded to version 8, all initiating events calculation types are changed from calculation type '1' to calculation type 'N'. Scenarios compare results in a project to prove that the 'N' calculation type upgrade has not changed results.	Pass
Test 72	RASP Common Cause Failure (CCF) validation	This test compares results in a project to prove that the RASP-CCF 'R' calculation type upgrade works properly and provides expected results with both rolled-up and expanded output. These test results will need to be verified by an expert in the RASP CCF field. Make sure the flag set adjustment is validated.	Pass
Test 73	External Event Models - Solve Event Trees	This test links external event model event trees, generates basic event data (with no change sets), solves (with cut set probability cutoff) and quantifies sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.	Pass
Test 74	Shutdown Models - Solve Event Trees	This test links shutdown model event trees, generates basic event data (with no change sets), solves (with cut set probability cutoff) and quantifies sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.	Pass
Test 75	Workspace model independence	This test ensures that databases move properly into the workspace and that workspace information remains independent from other workspaces and do not impact the base model.	Pass

Test #	Test Name	Description	Pass or Fail?
Test 76	Repetition of critical calculations over N times	This test ensures that cut set solving and recovery when done N times in a row calculate the same cut sets and quantification values for standard analysis. Do for Standard analysis, Event and Condition Analysis, Significance Determination Process , and General Analysis.	Pass
Test 77	Significance Determination Process -LERF multiplier calculations	This test ensures that Significance Determination Process -LERF multipliers are being used properly to calculate Screening LERF values.	Pass
Test 78	Accident Sequence Matrix - Solve Event Trees	This test links event trees after an Accident Sequence Matrix has been loaded, generate basic event data (with no change sets), solve (with cut set probability cutoff) and quantify sequence minimal cut sets, and recovery rules. The current case min cut upper bound, base case min cut upper bound, and cut set totals are verified for each sequence.	Pass
Test 79	Multiple pass algorithm test (True and 1.0) (See #1 ATP input draft)	This test sets one or more basic events to 1.0 and validates the generated cut sets to ensure proper cut set creation. It will also set one or more basic events to TRUE and validate the generated cut sets to ensure proper cut set creation.	Pass
Test 80	Multiple pass algorithm test (False and Ignore) (See #2 ATP input draft)	this test sets one or more basic events to False and validates the generated cut sets to ensure proper cut set creation. It will also set one or more basic events to Ignore and validate the generated cut sets to ensure proper cut set creation.	Pass
Test 81	Min-Max test on Demo-EE model for Event and Condition Analysis and General Analysis interfaces (See #3 ATP input draft)	This test will quantify all the DEMO-EE sequences using the min/max method to ensure the validity of the frequencies. One scenario will test it for an Event and Condition Analysis condition assessment, another scenario will test it for an Event and Condition Analysis Initiating Event Assessment, and another scenario will test it for a General Analysis.	Fail (will test with a simpler project)
Test 82	Single pass algorithm tests on Event and Condition Analysis and General Analysis (See #4 ATP input draft)	This test will exercise Event and Condition Analysis and General Analysis workspace analysis in the following areas: the number of sequences; total CCDP; total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified. Automated steps performed for condition assessments include basic event generation with change sets; and generation, quantification, and recovery of cut sets. One of the scenarios will test the Event and Condition Analysis interface and another will test the General Analysis interface.	Pass

Test #	Test Name	Description	Pass or Fail?
Test 83	Cross-referencing is validated (See #8 ATP input draft)	This test will exercise the various cross referencing capabilities.	Pass
Test 84	Verify database recovery works (See #11 ATP input draft)	This test will exercise the database recovery capabilities.	Pass
Test 85	Verify event tree/fault tree transfers function correctly (Manual tests) (See #12 ATP input draft)	This test will exercise the transfer functions.	Pass
Test 86	Gather End States on a demo model with multiple phases	This test will exercise the end state gathering on a demo model with multiple phases.	Fail (cut sets can be solved for different phases, but end state not assigned to them)
Test 87	Large Early Release Frequency (LERF) model functionality	This test opens an existing LERF model and exercises the Standard Analysis interface. The LERF model will be one of the models created in SAPHIRE 7 to calculate LERF results (Peach Bottom or Surry).	Pass (the Surry S_LERF model passes)
Test 88	Event Tree	This tests Event Tree Creation in a new project by building in the Standard Analysis interface a demonstration sized model with 3 phases and two model types from scratch and save the new project. A software developer will review these results initially. Subsequent tests will compare against verified output.	Pass
Test 89	Menu Navigation	This tests the Significance Determination Process and the Event and Condition Analysis interfaces. The buttons for moving back and forth between screens, canceling, and saving will be tested.	Pass
Test 90	Fault Tree View Expanded	These verifies that in the Standard Analysis interface the fault tree feature View Expand is working.	Pass
Test 91	Workspace to Standard Analysis Interface Independence	This test creates and saves new Significance Determination Process, Event and Condition Analysis, and General Analysis workspaces. Standard Analysis should never see any impact from workspace activity other than noting any saved workspaces in the workspace window. Test on Significance Determination Process, Event and Condition Analysis, General Analysis.	Pass

Test #	Test Name	Description	Pass or Fail?
Test 92	Standard Analysis Interface to Workspace Independence	This test runs a change set in the Standard Analysis interface and visually verify that running this change set does not alter existing workspaces. Test on Significance Determination Process, Event and Condition Analysis, General Analysis workspaces.	Pass
Test 93	Workspace to Workspace Independence	This tests the addition of a new workspace or editing existing workspaces do not impact other workspaces. Tests should verify that changes made to Standard Analysis after the creation of a workspace should not reflect that change. Test on Significance Determination Process, Event and Condition Analysis, General Analysis workspaces.	Pass
Test 94	Project to Project Independence	This test verifies that opening up a project does not include anything from a previously opened project database. Examine database to ensure previous database (different from the one just opened) information is not present.	Pass
Test 95	Workspace to Workspace Independence	This test verifies that information from a previous case run in a workspace is not showing up in the next created case.	Pass
Test 96	Integrated Models	This test verifies that Demo-EE model produces expected results in the Standard Analysis, the Event and Condition Analysis, and the General Analysis interfaces.	Pass
Test 97	Event Tree Linking and Unlink in integrated models (External Events and Shutdown)	This test verifies that the event tree linking and unlinking functions work properly for selected integrated models with External Events and Shutdown information.	Pass
Test 98	Verification of cut set view path	This test verifies all cut set path features work. The initial test will be visually verified with reports produced. Subsequent tests will compare reports to verified reports. Especially test this capability in the advanced Significance Determination Process report per cut set.	Pass
Test 99	Verification Rule layering works	This test verifies linkage, post- processing, partitioning, and slice rule layering works.	Fail
Test 100	Verification Rule Nesting works	This test verifies linkage, post- processing, partitioning, and slice rule nesting works.	Fail
Test 101	All reports produce expected reports	This test verifies the production of all the reports available in all the workspaces and interfaces. Initially all the reports will be produced and validated and then all future tests will compare freshly produced reports to the validated ones.	Pass

Test #	Test Name	Description	Pass or Fail?
Test 102	Significance Determination Process Interface testing of basic event changes	This test exercises Significance Determination Process basic event testing. Ensure that choosing True" performs a CCF probability calculation correctly and choosing "True" automatically handles T&M basic events correctly.	Pass
Test 103	Significance Determination Process Interface testing of Figure III-D (Change in delta CDF as a function of duration) point estimate checks	This test exercises Significance Determination Process workspace output. Figure III-D Change in delta CDF as a function of duration) is based upon a one hour value expanded to a full year outage. Test to determine that these point estimates are correct.	Pass
Test 104	Event and Condition Analysis uncertainty calculations	This test exercises Event and Condition Analysis workspace uncertainty calculations and corresponding graph of the Importance = CCDP - CDP.	Pass
Test 105	HRA-Report-Validate	This test exercises HRA Event output via reports.	Pass

NRC FORM 335 U.S. NUCLEAR REGULATORY COMMISSION

(2-89)
NRCM 1102,
3201. 3202

BIBLIOGRAPHIC DATA SHEET

1. REPORT NUMBER (Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)
NUREG/CR-7039, Vol. 6
INL/EXT-09-17014

2. TITLE AND SUBTITLE
Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8
Volume 6: Quality Assurance

3. DATE REPORT PUBLISHED
MONTH: MARCH
YEAR: 2011

4. FIN OR GRANT NUMBER
N6423

5. AUTHOR(S)
C. L. Smith, R. Nims, K. J. Kvarfordt

6. TYPE OF REPORT
Technical

7. PERIOD COVERED (Inclusive Dates)

8. PERFORMING ORGANIZATION - NAME AND ADDRESS (If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)
Idaho National Laboratory
Battelle Energy Alliance
P.O. Box 1625
Idaho Falls, ID 83415-3850

9. SPONSORING ORGANIZATION - NAME AND ADDRESS (If NRC, type "Same as above"; If contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

10. SUPPLEMENTARY NOTES
D. O'Neal, NRC Project Manager

11. ABSTRACT (200 words or less)
The Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8 is a software application developed for performing a complete probabilistic risk assessment using a personal computer running the Microsoft Windows™ operating system. SAPHIRE 8 is funded by the U.S. Nuclear Regulatory Commission (NRC). The role of the INL in this project is that of software developer and tester. This development takes place using formal software development procedures and is subject to quality assurance (QA) processes. The purpose of this document is to describe how the SAPHIRE software QA is performed for Version 8, what constitutes its parts, and limitations of those processes. In addition, this document describes the Independent Verification and Validation that was conducted for Version 8 as part of an overall QA process.

12. KEY WORDS/DESCRIPTORS (List words or phrases that will assist researchers in locating the report.)
SAPHIRE 8, software, reliability, risk, safety, PRA, quality assurance

13. AVAILABILITY STATEMENT
Unlimited

14. SECURITY CLASSIFICATION
(This page)
Unclassified
(This report)
Unclassified

15. NUMBER OF PAGES

16. PRICE

NRC FORM 335 (2-89)