INL/EXT-05-00655
Rev. 1

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Quality Assurance Manual

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

August 2008

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

Idaho National Laboratory

INL/EXT-05-00655
Rev. 1

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Quality Assurance Manual

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

August 2008

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 N6203

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, 11555 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

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) is a software application developed for performing a complete probabilistic risk assessment using a personal computer running the Microsoft Windows™ operating system. SAPHIRE is primarily 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 6 and 7, what constitutes its parts, and limitations of those processes.

FOREWORD

The U.S. Nuclear Regulatory Commission has developed the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) software 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.

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 can also 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 it includes 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 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. The versions currently in use (6 and 7) run in the Microsoft Windows® environment. A user-friendly interface, Graphical Evaluation Module (GEM), streamlines and automates selected SAPHIRE inputs and processes for performing event assessments.

SAPHIRE has also evolved with users' needs, and Versions 6 and 7 include new features and capabilities for developing and using larger, more complex models. For example, Version 7 can solve up to 2 million sequences and includes enhancements for cut set slicing, event tree rule linkage, and reporting options.

This NUREG-series report comprises seven volumes, which address SAPHIRE/GEM Versions 6 and 7. Volume 1, "Overview/Summary," gives an overview of the functions available in SAPHIRE and presents general instructions for using the software. Volume 2, "Technical Reference," discusses the theoretical background behind the SAPHIRE functions. Volume 3, "SAPHIRE Users' Manual," provides installation instructions and a step-by-step approach to using the program's features. Volume 4, "SAPHIRE Tutorial Manual," provides an example of the overall process of constructing a PRA database. Volume 5, "GEM/GEMDATA Reference Manual," discusses the use of GEM. Volume 6, "SAPHIRE Quality Assurance (QA) Manual," discusses QA methods and tests. Lastly, Volume 7, "SAPHIRE Data Loading Manual," assists the user in entering PRA data into SAPHIRE using the built-in MAR-D ASCII-text file data transfer process.

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

CONTENTS

EXECUTIVE SUMMARY

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.

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 and new features.
- Using a cyclical design process to prototype changes.
- Performing acceptance tests that the software must pass prior to official release.

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 made, 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 versions 6 and 7, 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. GEM module functionality

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.

ACRONYMS

GEM	Graphical Evaluation Module
INEEL	Idaho National Engineering and Environmental Laboratory
INL	Idaho National Laboratory
IRRAS	Integrated Reliability and Risk Analysis System
NRC	Nuclear Regulatory Commission
PC	Personal Computer
PRA	Probabilistic Risk Analysis
QA	Quality assurance
RAW	Risk Achievement Worth
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
TV&V	Testing, Verification, and Validation
V&V	Verification and Validation

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

Vol. 6 Quality Assurance Manual

1. INTRODUCTION

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).

Using SAPHIRE 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 representations 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. In particular, GEM implements many of the accident sequence precursor (ASP) program analysis methods. Using GEM, an analyst can estimate the risk associated with operational events very efficiently and expeditiously.

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 describe the key part of the process. Second, the report discusses 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 Current SAPHIRE QA Process

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). Within that document, Section 1 explains that the version 6 and 7 TV&V departs from earlier V&V efforts (for versions 4 and 5) by focusing on the development and execution of a set of automated test scripts. This 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. Note that

over the development cycle spanning from 1999 to the end of 2002 (four years), the INL released 18 versions of SAPHIRE 6.x for an average of one incremental release every 2.7 months. As the software matures, however, the release frequency tends to decrease.

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 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.

1.2.2 Acceptance Testing/Automated Testing

Prior to any official SAPHIRE release of versions 6 and 7, 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
- GEM module 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.

The SAPHIRE automated test suite was designed to verify core operations, such as generating current event data, and solving for cut sets. 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.

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.

Each new version of SAPHIRE undergoes beta testing before its release. 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.

In addition to the automated testing employed by the SAPHIRE TV&V, the development team utilizes a multi-faceted approach to testing. This approach, illustrated in **Figure 2**, 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 NUREG/CR-6688 and this report.

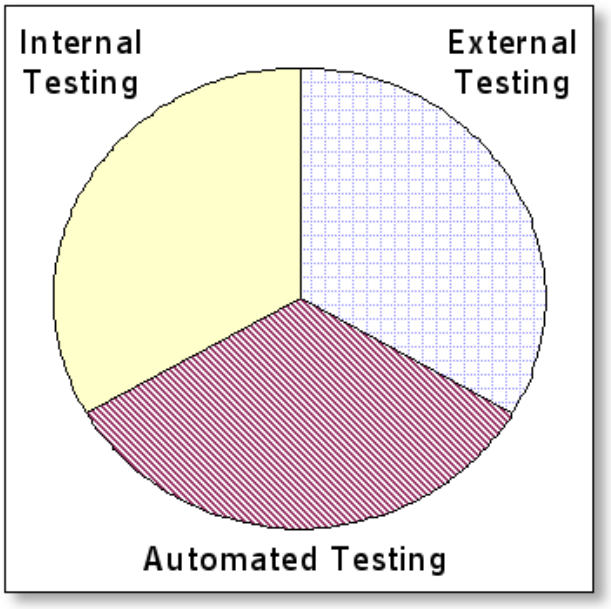


Figure 2. Types of testing used during the SAPHIRE development process.

1.2.3 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 an on-line user manual and technical reference manual. Individual changes to the software are not necessarily reflected in this documentation with each release. Many changes are not applicable to this level of the documentation, but some changes and new features do apply. Minor changes, such as wording changes in a screen shot, or removal of an obsolete feature, do not merit immediate inclusion; however, significant new features warrant timely addition. As priorities, time, and budget permit, when such new features are added to the software this documentation is revisited and updated.

1.2.4 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 Automated Testing section below.

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.

1.2.5 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.

2. QUALITY ASSURANCE PROCESSES

2.1 Tests Used in the SAPHIRE TV&V

The use of SAPHIRE in regulatory applications is extensive. Therefore, SAPHIRE is tested through various processes. Each new SAPHIRE version is beta tested to some degree before its release. Beta testers are analysts experienced with PRA methods and terminology and are typically familiar with earlier versions of SAPHIRE. The primary objective of the beta testing is to verify that the results produced by the new version are correct. The secondary objective is to ensure the software is user-friendly and functional. In addition, INL personnel receive feedback from users around the world. Hundreds of users rely on the calculations inherent in SAPHIRE for both risk and reliability calculations. New SAPHIRE releases are tested extensively by (a) comparing them with PRA models and results of earlier versions and (b) by loading new PRAs and comparing them with expected results. Given that different PRAs have been performed with different types of software, one can argue that SAPHIRE has been tested with an enormous number of test cases.

The test procedure dictates how the mechanics of the testing process is to take place. To perform the tests for the TV&V, test scripts and test databases to be used are stored on a network drive (at the INL) accessible by version control software. The version control software tracks all changes by author and time. Note that only one person is allowed to check out an item for modification at any one time. These personal copies are stored on a local machine for development and testing. Any completed changes are then submitted to the version control library with the name of the author, date, time, and a short description of the change. The version control software stores and marks the changed copy as the newest version but retains the old versions for historical purposes.

Individual test cases are designed to perform a specific analysis task, just as a SAPHIRE user might perform them. Each test case consists of one or more scenarios (e.g., modifying data, generating cut sets). These scenarios focus on a particular piece or variation of the test case analysis task. The complete set of tests and scenarios comprise the test suite, which is executed prior to release of each new version of SAPHIRE.

Prior to running the test suite, the latest, completed, and debugged scripts are checked out of the control library and compiled (by the testing software) into run-time form. The compiled suite of tests, along with the compressed (.zip format) database files and SAPHIRE, are transferred to the test machine on which the tests are to be run (if any changes to the scripts have been made since the last test run). This delivery mechanism allows the TV&V team to test SAPHIRE on a variety of computer platforms and operating systems. Currently, SAPHIRE is supported for the Microsoft Windows operating systems of Windows 98, Windows NT, Windows 2000, and Window XP. The SAPHIRE software should function properly under derivatives of these operating systems (e.g., Windows ME), but at this time, the TV&V has not evaluated these other operating systems.

The SAPHIRE test utilizes two different processes. The first process is the development of test scripts or batch files, which are DOS commands that run pre-determined macros. The macros are the second process that is used in the testing of the newly released versions of SAPHIRE. Both of these processes along with a simple example will be presented.

The script of batch files are DOS commands that set up the test that will be performed. There are two different types of scripts. The first one shown below is used to execute multiple scripts at once. This script file sets up the output by stating the date and time the test was ran along with what version of SAPHIRE was executed. This script will be considered as the overall test script. The individual lines are DOS commands that help create the output. The first line is an input (i.e., %1 = typed in file name [detail]), which is a file that will be created storing all of the output information. The second line is also a file that contains output information but is a summary report instead of a full detail report (i.e., %2 = typed in file name [summary]). The next group of lines is used to create the headers listed in the detail output report. These lines stamp the output with the date and time of analysis along with the version of SAPHIRE being ran. This information is placed in both the detail and summary report. The last group of lines will now call the individual script files used for specific evaluations. In the case shown the core damage frequency analysis will be performed using the database. The line is a DOS line, which has the core_damage_freq script executed with the output information being stored in %1 (detail file output) and %2 (summary file output) and then which database this script is to be executed. The individual script files will be discussed briefly since they are very similar.

```
Overall Test Script
if %1$==$ goto end
if %2$==$ goto end

c:
cd \Saphire7
echo SAPHIRE/GEM Test Suite Summary Report > %2
c:\Saphire7\qatools\datetime  "DATE & TIME :" %2
c:\Saphire7\qatools\fversion  c:\Saphire7\tools\saphwin.exe %2
echo>> %2
echo> %1

rem CDF analysis
call scripts\core_damage_freq %1 %2 byrn_2qa
etc.
```

The individual script files can be ran as stand-alone or via an overall test script. The individual test scripts are similar except they execute the macros, which tell SAPHIRE what type of analysis is to be performed. The first two lines are the same for the individual test script as for the overall test script. The third line, however, represents the database to be evaluated (%3 = database). The next group of lines is used to create the folder, which the database will be placed and unzipped. Then it executes SAPHIRE and calls the macro, which has the details of the specific analysis. Once the macro has performed its specific analysis the results are dumped into the detail output file (%1) and summary output file (%2). The last line is used to compile all of the outputs together in order to create one large detail file and one large summary file.

```
Individual Test Script
if %1$==$ goto end
if %2$==$ goto end
if %3$==$ goto end

md c:\Saphire7\%3
del c:\Saphire7\%3\*.* /q
c:\Saphire7\qatools\unzip  o c:\Saphire7\database\%3.exe  d c:\Saphire7\%3
copy c:\Saphire7\results\%3\qa*.rpt c:\Saphire7\%3
```

call c:\Saphire7\tools\saphwin.exe i386 PROJECT=c:\saphire7\%3
MACRO=c:\saphire7\macros\Core_Damage_Freq_%3.mac
DETAIL=core_damage_freq_%3.rpt
copy %1 + c:\Saphire7\%3\core_damage_freq_%3.rpt %1
c:\Saphire7\qatools\lastline c:\Saphire7\%3 c:\Saphire7\%3\core_damage_freq_%3.rpt %2
:end

The macros are used to perform specific analyses. The following will provide a brief overview of the macros. The macros utilize key words or verbs. The verbs or key words are designed to execute certain functions within SAPHIRE to perform the specific analysis. The following macro that will be discussed is used to solve the fault trees and event tree accident sequences of the specified project database. The macro is core_damage_freq.mac. The macro will be dissected in order for better understanding.

First, the macro sets up the analysis. The first line states that no prompt is required prior to starting the analysis (i.e., SAPHIRE will just move down to the execution process instead of waiting for a manual input).

<initial prompt>no</initial prompt>

This part is a comment bracket, which enables the analyst to identify the type of analysis this macro is going to perform and any other pertinent information.

<comment>
TEST CASE NAME: Core Damage Frequency
TEST SCRIPT FILE NAME: Core_Damage_Freq_PWR.mac

GENERAL DESCRIPTION OF WHAT IS VERIFIED:
This test case compares the sequence current case CDF against SAPHIRE version 6 base case results. This test is not plant specific.

NAME OF APPLICATION UNDER TEST: SAPHIRE 7.0

TEST CASE PURPOSE:
REQUIREMENT(S) VERIFIED: TBD

TEST 01 Solve Fault Trees Fault Tree Probability Results
TEST 02 Solve Sequences Core Damage Frequency Results

TEST CASE ABSTRACT OF TECHNIQUES USED TO TEST THE FEATURE:
The automated tests described herein are grouped to run consecutively.
OTHER FILES REQUIRED TO RUN TEST CASE: None.
</comment>

This part provides a description of the particular scenario that is going to be evaluated. The scenario for this case is %P-01, where %P represents the particular database (i.e., Byrn_2qa) then provides the description of the test (i.e., Solve Fault Trees).

<scenario>
<start>
<name>%P 01</name>

```
  <description>Solve Fault Trees</description>
 </start>
</scenario>
```

The fault tree menu is now executed by using the key word or verb <fault tree>. All of the fault trees are marked via the "*" operator, then they are solved at a truncation of 1.0E-16. Once all of the fault trees have been solved, a base case update in the random calculation type is performed (key word <base case update>, <analysis>random</analysis>). The results are then sent to a file with the name specified (i.e., ft_current_vs_base.rpt). This output is then compared to a quality assured set of results to make sure this version of SAPHIRE that is being tested matches the results of a quality assured version of SAPHIRE. Then the fault tree menu option is exited and the scenario is ended.

```
<fault tree>
 <unmark></unmark>
 <mark mask>*</mark mask>
 <solve>
  <truncation>1.0E 16</truncation>
 </solve>
 <base case update>
  <analysis>random</analysis>
 </base case update>
 <report>
  <type>results</type>
  <sub type>current base</sub type>
  <file name>ft_current_vs_base.rpt</file name>
 </report>
 <compare file>
  <input 1>ft_current_vs_base.rpt</input 1>
  <input 2>qa_ft_current_vs_base.rpt</input 2>
  <output>compare.rpt</output>
 </compare file>
 <report>
  <type>results</type>
  <sub type>current only</sub type>
  <file name>ft_current_only.rpt</file name>
 </report>
 <compare file>
  <input 1>ft_current_only.rpt</input 1>
  <input 2>qa_ft_current_only.rpt</input 2>
  <output>compare.rpt</output>
 </compare file>
</fault tree>
<scenario><end></end></scenario>
```

The next part of the macro provides a description of the particular scenario that is going to be evaluated. The scenario for this case is %P-02, where %P represents the particular database then provides the description of the test (i.e., core damage frequency test).

```
<scenario>
 <start>
  <name>%P 02</name>
  <description>Core Damage Frequency Test</description>
 </start>
</scenario>
```

This section tells SAPHIRE to go into the Change Set menu and make sure there are no change sets marked, then generate the basic event data. Lastly, the process is completed using the </change set> key word.

```
<change set>
 <unmark></unmark>
 <generate></generate>
</change set>
```

The last line of every macro is the "exit program" verb. This command causes the macro to exit from SAPHIRE in order for another macro to be executed.

```
<program exit></program exit>
```

The above macro provided only a brief description of how the verbs or key words work in the macros. However, all of the other menus and actions that can be performed by SAPHIRE (i.e., end state evaluations, importance measures, GEM evaluations, etc) can be created using the same format. In general, state the starting key word or verb (<end state>), then add the type of evaluation required (<solve> [i.e., gather cut sets]), then end the process by adding "/" to the verb (</end state>).

By developing the key words or verbs into a SAPHIRE macro, the software can be tested for efficient and direct version verification. The automated test suite uses embedded software hooks in the application-programming interface (API) to allow the application code to run the test macros. The original test sequences (from SAPHIRE version 5) were translated into the macro language and then rerun on the current SAPHIRE software release to ensure results matched the pre-macro results. The pre- and post-macro results were independently verified and validated by a PRA analyst.

In addition to specific test scenario data, the user identification of the person running the test; the version of SAPHIRE being tested; and the version of the operating system are automatically recorded. Final acceptance of any documentation, code, and test results is considered complete when all parties sign off on the completed change.

The automated test software generates two documents: a summary report and a detail report, as it executes the tests. The report lists the test identification number, a description, and an overall pass/fail indicator. A test is failed if even a single value in one sub-test is incorrect. The detail report displays a more thorough description of the steps taken, the results obtained, the expected results, and deviations, if any. As the code developers run the test suite, any discrepancies are noted and corrected prior to release of a new version.

Automated testing activities are used to provide faster, better, and more efficient assessment of the SAPHIRE code. Other test activities include the individual tests conducted by the developer of specific module(s). Individual test cases are designed to perform a specific analysis task. Each test case consists of one or more scenarios. These scenarios focus on a particular variation of the test case analysis task. A complete set of tests/scenarios comprise the test suite, which is run for each new version of SAPHIRE.

Before test scripts are created, the salient features of the software to be tested must first be identified. Identification of the SAPHIRE features to be tested begins by outlining the major functions performed in a PRA. These functions are then overlaid onto specific SAPHIRE features. Applicable PRA functions include cut set generation and quantification; uncertainty analysis; and importance measures. Input is solicited and received from experienced PRA users to expand and refine the list. From the list, SAPHIRE features are examined to determine importance and if they testable.

Once the important SAPHIRE features are identified, tests were generated that would evaluate each feature are selected. These tests may have more than one type of analysis approach, since it is possible within PRA (and SAPHIRE also) to solve some problems in more than one way. For example, sequence cut sets could be determined by solving sequence logic explicitly or by combining pre-existing fault tree cut sets.

For each test result in the suite (see Appendix C), the first line of the test result identifies the test ID and description along with the time at which the particular test was started. This is illustrated below in the sample test result (e.g., SURRY-50-05). After the identifier line, the steps processed by the test are shown. In the example below, the SURRY-50 sequences are solved using a truncation of 1E-9/yr and then recovery rules are applied. The cut sets are run through a cut-set update. Then, the test gathers end-state cut sets via the partition rules (again with 1E-9/yr truncation). These end-state cut sets are updated. Lastly, the results are compared against the stored “correct” results for the end states of AD5, AD6, AH1, and S2D1. If the results match the “correct” results, a “pass” is indicated, otherwise a “failed” would be indicated. Then, the time of test completion is recorded.

SURRY-50-05 Scenario: Check End State Cut Sets started at 12:48:28 AM
Sequences solved with prob cut off (1.0E-09) and with recovery
Sequence cut sets updated
End States gathered by cut set partition with prob cut off (1.0E-09)
End State cut sets updated

END STATE CUTSET RESULTS:
AD5 pass
AD6 pass
AH1 pass
S2D1 pass
Scenario: Check End State Cut Sets completed at 12:50:05 AM

While the tests and approved criteria address a large part of the calculation functionality within SAPHIRE, the tests do not cover 100% of SAPHIRE's 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 (i.e., "post-processing") by manually editing them, using "recovery rules," using the "prune" option, and performing a cut set update. However, the test suite does test the most commonly used mechanisms of performing tasks in SAPHIRE – these PRA tasks and their associated tests are listed in Table 1. In this table, the test number, its name, the PRA area/function tested, the SAPHIRE option that is exercised, and the test model(s) that are used are listed.

Table 1 Tests where specific PRA features are verified

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-01	Solve Fault Trees Fault Tree Probability Results.	Generate current event data	No change set data	All
Test-02	Solve Sequences Core Damage Frequency Results.			
Test-05	Transient initiator with no other failures.			
Test 06	Small LOCA initiator with no other failures.			
Test-07	Steam Generator initiator with no other failures.			
Test-08	Grid-related LOOP initiator with no other failures.			
Test-09	Plant-centered LOOP initiator with no other failures.			
Test-10	Severe Weather LOOP initiator with no other failures.			
Test-11	Extreme Severe Weather LOOP initiator with no other failures.			
Test-13	Project Uncertainty.			
Test-14	Log Normal Distribution using MCS.			
Test-15	Normal Distribution MCS.			
Test-16	Beta Distribution MCS.			
Test-17	Chi-squared Distribution MCS.			
Test-18	Exponential Distribution MCS.			
Test-19	Uniform Distribution MCS.			
Test-20	Gamma Distribution MCS.			
Test-21	Maximum Entropy Distribution MCS.			
Test-23	Seismic Log Normal Distribution MCS.			
Test-24	Constrained Non-informative Distribution MCS.			
Test-25	Log Normal Distribution using LHS.			
Test-26	Normal Distribution LHS.			
Test-27	Beta Distribution LHS.			
Test-28	Chi-squared Distribution LHS.			
Test-29	Exponential Distribution LHS.			
Test-30	Uniform Distribution LHS.			
Test-31	Gamma Distribution LHS.			
Test-32	Maximum Entropy Distribution LHS.			
Test-34	Seismic Log Normal Distribution LHS.			
Test-35	Constrained Non-informative Distribution LHS.			
Test-36	Histogram Distribution MCS.			
Test-37	Histogram Distribution LHS.			
Test-38	Gather End States.			
Test-39	End State Single/Group Uncertainty MCS.			
Test-40	End State Single/Group Uncertainty LHS.			
Test-42	Link Level 1 Event Trees (small event trees).			
Test-43	Partition Sequence Cut Sets.			
Test-44	Link PDS Event Trees (large event trees)..			
Test-45	Fault Tree Importance Measures.			
Test-46	Sequence Importance Measures.			
Test-47	Sequence Group Importance Measures.			
Test-48	End State Importance Measures.			
Test-49	End State Group Importance Measures.			
Test-03	Condition Assessment - MFW unavailable for 72 hours.	Generate current event data	Single changes	DEMO, SURRY-50
Test 04	Emergency Diesel Generator out of Service for 3 months.			
Test-12	Transient initiator with AFW failed.			
Test-22	Dirichlet Distribution MCS.			
Test-33	Dirichlet Distribution LHS.			
Test-50	Single Change Set on Compound Event.			
Test-50	Single Change Set on Compound Event	Generate current event data	Single changes	Wolf Creek, Peach Bottom

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-51	Change Set Processing- Class. Class change - all events, probability 0.1 Class change - ?-MOV-1 events, probability 0.5 (a subset)	Generate current event data	Class changes	DEMO, SURRY-50
Test-52	Change Set Processing-Marked Order Marked change sets from scenarios 1, 2, 3 (marked in that order)	Generate current event data	Marked order	DEMO, SURRY-50
Test-41	Cut Set Verification Cut Set Verification Solve Sequences and Monte Carlo uncertainty calculations. Cut Set Verification Solve Fault Trees, Sequences and end states.	Fault tree cut set generation	With flag sets	COM-PEAK, SURRY-50, SEQUOIA
Test-01 Test-41 Test-53	Solve Fault Trees Fault Tree Probability Results Cut Set Verification Basic Events Load / Extract Fault Tree Load / Extract	Fault Tree cut set generation	Without flag sets	SPAR, ** COM-PEAK, SURRY-50, CR3
Test-41	Cut Set Verification	Sequence cut set Generation	With flag sets	All
Test-02 Test-13 Test-41 Test-42	Solve Sequences Core Damage Frequency Results Project Uncertainty Cut Set Verification	Sequence cut set Generation	Without flag sets	Multiple
Test-38 Test-41 Test-44	Gather End States Cut Set Verification Link PDS Event Trees (large event trees).	Gather sequence cut sets into end states	By sequence	BV2-5, SURRY-50, COM-PEAK, S_LERF
Test-43	Partition Sequence Cut Sets.	Gather sequence cut sets into end states	By Cut Set	S_LERF
Test-14 Test-21 Test-23 Test-24	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution Constrained Non-informative Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-14 Test-21 Test-23 Test-36	Log Normal Distribution using MCS Maximum Entropy Distribution Seismic Log Normal Distribution Histogram Distribution-MCS	Uncertainty of fault tree distributions	Monte Carlo sampling	TSTU
Test-25 Test-26 Test-27 Test-28 Test-29 Test-30 Test-31 Test-32 Test-33 Test-34 Test-35 Test-37	Log Normal Distribution using LHS Normal Distribution using LHS Beta Distribution LHS Chi-squared Distribution LHS Exponential Distribution LHS Uniform Distribution LHS Gamma Distribution LHS Maximum Entropy Distribution LHS Dirichlet Distribution LHS Seismic Log Normal Distribution LHS Constrained Non-informative Distribution LHS Histogram Distribution-LHS	Uncertainty of fault tree distributions	Latin Hypercube sampling	TSTU
Test-22 Test-24	Dirichlet Distribution MCS Constrained Non-informative Distribution MCS	Sequence uncertainty analysis	Monte Carlo sampling	TSTU
Test-38	Gather End States	Gathering of End States		BV2-5
Test-39	End State Single/Group Uncertainty MCS	End State uncertainty analysis	Monte Carlo sampling	BV2-5
Test-40	End State Single/Group Uncertainty LHS	End State uncertainty analysis	Latin Hypercube sampling	BV2-5
Test-45	Fault Tree Importance Measures Fault Tree Fussell-Vesely Importance (ratio) Fault Tree Birnbaum Importance (Interval or difference) Fault Tree Uncertainty Importance	Importance measures	Fault trees	DEMO
Test-46	Sequence Importance Measures Sequence Fussell-Vesely Importance (ratio). Sequence Birnbaum Importance (interval or difference). Sequence Uncertainty Importance.	Importance measures	Sequence	DEMO
Test-47	Sequence Group Importance Measures Sequence Group Fussell-Vesely Importance (ratio). Sequence Group Birnbaum Importance (interval or difference). Sequence Group Uncertainty Importance.	Importance measures	Sequence Group	BV2-5
Test-48	End State Importance Measures End State Fussell-Vesely Importance End State Birnbaum Importance End State Uncertainty Importance	Importance measures	End State	HISNO, BV2-5

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-49	End State Group Importance Measures End State Group Fussell-Vesely Importance End State Group Birnbaum Importance End State Group Uncertainty Importance	Importance measures	End State Group	BV2-5
Test-41 Test-53	Cut Set Verification Basic Events Load / Extract Fault Tree Load / Extract	Cut Set Update	Fault trees	SURRY-50, COM-PEAK, CR3
Test-13 Test-41	Project Uncertainty Cut Set Verification	Uncertainty analysis		
Test-41	Cut Set Verification	Cut Set Update	Fault trees	SURRY-50, COM-PEAK, CR3
Test-41 Test-53	Cut Set Verification Basic Events Load / Extract Fault Tree Load / Extract	Fault tree cut set Recovery	Auto-recover option	SURRY-50, COM-PEAK, CR3
Test-02 Test-13 Test-41	Solve Sequences Core Damage Frequency Results Project Uncertainty Cut Set Verification	Sequence cut set Recovery	Auto-recover option	SURRY-50, COM-PEAK
Test-43	Partition Sequence Cut Sets.	Sequence cut set Partitioning	Batch apply option	S_LERF
Test-42	Link Level 1 Event Trees (small event trees).	Link Small event tree (logic)	Linkage Rules	S_LERF
Test-44	Link PDS Event Trees (large event trees).	Link Large event tree (cut sets)	Create cut sets option	S_LERF
Test-41	Cut Set Verification	Fault Tree logic	Alpha-numeric logic editor	SURRY-50, COM-PEAK
Test-41	Cut Set Verification	Fault Tree logic	Graphical editor	SURRY-50, COM-PEAK, CR3
Test-54	Fault Tree Utilities: Auto Page/Solve Fault Tree Utilities: Cut Sets to End State	Fault Tree Logic	Pager	CR3
Test-42	Link Level 1 Event Trees (small event trees)	Event tree logic	Graphical editor	S_LERF
All tests		Project version controll	Version Upgrade	All Models
Test-05 Test-06 Test-07 Test-08 Test-09 Test-10 Test-11 Test-12	Transient initiator with no other failures. Small LOCA initiator with no other failures. Steam Generator initiator with no other failures. Grid-related LOOP initiator with no other failures. Plant-centered LOOP initiator with no other failures. Severe Weather LOOP initiator with no other failures. Extreme Severe Weather LOOP initiator with no other failures. Transient initiator with AFW failed.	Initiating Event Assessments	Delete	All Models

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-05 Test-06 Test-07 Test-08 Test-09 Test-10 Test-11 Test-12	Transient initiator with no other failures. Small LOCA initiator with no other failures. Steam Generator initiator with no other failures. Grid-related LOOP initiator with no other failures. Plant-centered LOOP initiator with no other failures. Severe Weather LOOP initiator with no other failures. Extreme Severe Weather LOOP initiator with no other failures. Transient initiator with AFW failed.	Initiating Event Assessments	Add	All Models
Test 03 Test-04	Condition Assessment - MFW unavailable for 72 hours Emergency Diesel Generator out of Service for 3 months	Condition Assessments	Delete	All Models
Test 03 Test-04	Condition Assessment - MFW unavailable for 72 hours Emergency Diesel Generator out of Service for 3 months	Condition Assessments	Add	All Models
Test 03 Test-04	Condition Assessment - MFW unavailable for 72 hours Emergency Diesel Generator out of Service for 3 months	Condition Assessments	Add events to Assessment	All Models
Test-03 Test-04	Condition Assessment - MFW unavailable for 72 hours Emergency Diesel Generator out of Service for 3 months	Condition Assessments	Process	All Models
Test-53	Basic Events Load / Extract Fault Tree Load / Extract	Fault Trees	Load/Extract	CR3
Test-53	Basic Events Load / Extract Fault Tree Load / Extract	Basic Events	Load/Extract	CR3
Test-54	Fault Tree Utilities: Auto Page/Solve Fault Tree Utilities: Cut Sets to End State	Fault Trees	Modify/Delete	CR3
Test-55	Link Level 1 Event Trees (small event trees). Link Level 1 Event Trees: Solve w/ Flag Sets Solve Sequence Cut Sets w/ no Flags.	Level 1 Event Tree Linking	Linkage rules	Wolf Creek 302, Peach Bottom 302, SIMPLE-FT
Test-56	End-State Gathering	End States	End State Gathering	S_LERF (by rules), Beaver Valley (by names)
Test-57	Compound Event Plug ins	Common cause module, Utility module (i.e,. add, multiply), Load-capacity	Fault Tree, Compound Event Plug-in	SIMPLE-FT (PLUG-IN-FT)
Test-58	Base Case Updates	Base Case Update	Base Case Update	All SPAR 2Q, 3i models
Test-59	Calculation Types & N of M Gates Calculation Type True, N of M Gates Calculation Type False, N of M Gates Calculation Type Ignore, N of M Gates	Calculation types	True, 1,3,5,7, False, 1,3,5,7 Ignore, 1,3,5,7 Use of AND gates, then OR gates.	SIMPLE-FT
Test-60	Change Sets (place holder for a future test)		Change Sets	TBD
Test-61	Uncertainty analysis		Uncertainty Distributions	TSTU, SUR40, Wolf Creek, Peach Bottom, SIMPLE-FT

Test Number	Test Name	PRA Area	SAPHIRE option	Test Models
Test-59	Calculation Types & N of M Gates Calculation Type True, N of M gates Calculation Type False, N of M gates Calculation Type Ignore, N of M gates	N of M Gates	Use of all inputs	SIMPLE-FT
Test-59	Load Capacity Test Calculation Type True, N of M Gates	Sequence generation	Sequence Generation	SIMPLE-FT (BE-LOAD-CAPACITY)
Test-64	Common-cause failure	Common Cause Failures, Basic Events with change sets	Common Cause Plug-ins	Wolf Creek, Peach Bottom 3
Test-65	Event Transformations (place holder for a future test)	Basic Events, Sequence Analysis	Transformations	TBD
Test-66	Wrong Results (a false positive to ensure the error flag for testing is functioning properly)	Results verification	Results verification	DEMO

2.2 QA Processes Used During the SAPHIRE Development

2.2.1 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 maintenance phase of the software development lifecycle, software development procedures and supporting company 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 Test Verification & Validation (TV&V) plan. These test procedures demonstrate the adherence to the requirements specified in the NRC forms.
- Recording and implementing lessons learned

2.2.2 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 project manager 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 project manager.

2.2.3 Documentation Purpose

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 a 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

Documentation for specifications, such as a Requirements Specification and Detailed Design Specification, are not formally generated. Guidance and requests for new functionality are received from the NRC via an alternate mechanism. Contract documents have served as the driving documentation for specification of software requirements and have not required the need for formal documentation, primarily because the SAPHIRE product is now in maintenance mode. The contract documents have provided all the necessary guidance for implementing technical requirements for new features and bug fixes. Typically, very little change to the detailed design of the software is affected by the addition of new features. As such, all requirements and the code designs needed for implementing those requirements have been verified and validated through the use of the SAPHIRE automated testing process, the TV&V plan, and reviewed by the product team to provide and ensure the quality of the software release.

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.

2.2.4 Testing, Verification, and Validation

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 TV&V plan is developed to make sure that all requirements are implemented and those new features do not affect existing code functionality or design. The TV&V is a consolidated document used for tracking the software development, testing and implementation and explicitly identifies the new features implemented for each release of the software as well as the automated test results, including regression tests, to ensure the software is complete, consistent, and correct. The SAPHIRE product development team uses the TV&V to track, verify and validate requirements to ensure that all requirements are implemented and that all requirements are included in the automated test scripts and test results. The TV&V plan is updated for each release of SAPHIRE by the development team by the performance of the following steps:

- Prepare the TV&V plan

- Determine the areas required for testing, including regression testing
- Develop new test cases based upon the development of a test model that includes the identification of available PRA obtained from the PRA database
- TV&V model testing which encompasses the identification of base-case or nominal results for each test case
- Documenting the test results, conclusions, and actions to correct any failures discovered during the automated testing process

Prior to any official SAPHIRE release, the software is run through a series of automated test procedures. These tests run SAPHIRE through calculation exercises in order to compare the output to expected results. This ensures that given a static input PRA file, the risk or reliability results from SAPHIRE will be consistent. These tests are developed by initially identifying the critical tasks performed in a PRA. These tasks are then mapped to the SAPHIRE functions that perform these tasks. 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
- Graphical evaluation module (GEM) functionality

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.

2.2.5 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.

2.2.6 QA Standards, Practices, and Conventions

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 TV&V 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.

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) T*esting, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0*, NUREG-CR/6688, September.

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	Feature Description	Version 6.x	Version 7.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
B	**Cut Set Generation for Fault Trees and Event Trees**	X	X
C	**Cut Set Gathering for Sequence and End State Cut Sets**	X	X
D	**Cut Set Partitioning via rules**	X	X
E	**Cut Set Sorting**		
E.1	By individual basic events	X	X
E.2	By probability		X
E.3	By rules		X
F	**Cut Set Post Processing (Recovery Rules)**		
F.1	Event tree sequences	X	X
F.2	Fault trees	X	X
G	**Change Sets (modifying basic events for an analysis)**		
G.1	Single event selection	X	X
G.2	Multiple event selection	X	X
G.3	Group event selection	X	X
G.4	Workspace area		
H	**Flag Sets (setting basic events to True of False)**		
H.1	Static (predefined) flag sets	X	X
H.2	Dynamic (rule based) flag sets	X	X
I	**Cut Set Quantification Methods**		
I.1	Minimal Cut Set Upper bound (min-cut)	X	X
I.2	Min-Max (exact, using inclusion/exclusion principle	X	X
I.3	Rare Event	X	X
I.4	Split Fraction (Sequences only)	X	X
J	**Cut Set Analysis**		
J.1	Cut set generation – cut sets solved, gathered, with truncation by size or probability, auto recovery	X	X
J.2	Cut set path tracing through logic model	X	X
J.3	Cut set comparison between two cases	X	X
J.4	Cut set comparison including probability changes		
J.5	Fault tree	X	X
J.6	Event tree sequences	X	X
J.7	End states	X	X
K	**Basic Events**		
K.1	Basic event correlation designation	X	X

Item	Feature Description	Version 6.x	Version 7.x
K.2	Basic event templates (reuse of a single event)	X	X
K.3	Compound events (plug in modules)		X
K.3.1	Common-cause alpha-factor module	X	X
K.3.2	Common-cause alpha-factor (staggered) module	X	X
K.3.3	Common-cause beta-factor module	X	X
K.3.4	Common-cause multiple Greek letter module	X	X
K.3.5	Loss-of-offsite power frequency and recovery module		X
K.3.6	Time Series module	X	X
K.3.7	General purpose utility module	X	X
K.3.8	Load-capacity module		X
K.3.9	Flow acceleration corrosion pipe module	X	X
K.3.10	User defined module		
K.4	Failure probability on demand	X	X
K.5	Failure probability to run	X	X
K.6	Value input (for any value)		X
K.7	Failure probability to run w/ repair	X	X
K.8	Failure probability to run	X	X
K.9	House event True (Prob = 1.0) i.e. failed	X	X
K.10	House event False (Prob = 0.0) i.e. success	X	X
K.11	House event Ignore	X	X
K.12	Human Factor Event		X
K.13	Fault tree Min Cut Upper Bound Value	X	X
K.14	End State Min Cut Upper Bound Value	X	X
K.15	Seismic screening using user-specified ground acceleration value	X	X
K.16	Seismic screening using hazard curve	X	X
L	**Importance Measures**		
L.1	Fussell-Vesely importance measure	X	X
L.2	Birnbaum importance measure	X	X
L.3	Risk increase ratio importance measure	X	X
L.4	Risk reduction ratio importance measure	X	X
L.5	Risk increase interval importance measure	X	X
L.6	Risk reduction interval importance measure	X	X
L.7	Group importance measure	X	X
L.8	Uncertainty determination on importance measures		X
M	**Model Creation**		
M.1	Seismic, fire and flooding transformation capability	X	X
M.2	Fault tree logic editor	X	X
M.3	Fault tree graphical editor	X	X
M.4	Event tree graphical editor	X	X
N	**Model Creation Load / Extract Data Models (MAR-D)**		
N.1	All data and file types concurrently		X
N.2	Project files (primary descriptions, attributes, recovery rules, fault tree recovery rules, partition rules, primary text)	X	X
N.3	Project files (alternate description, alternate text)		X
N.4	Attributes (primary attributes)	X	X
N.5	Attributes (all attributes, alternate attributes)		X

Item	Feature Description	Version 6.x	Version 7.x
N.6	Basic event files (description, rate information, attributes, transformations)	X	X
N.7	Basic event files (alternate description, text, alternate text, compound events)		X
N.8	Fault tree files (description, logic, graphics, cut sets, attributes, recovery rules, primary text, PID diagrams)	X	X
N.9	Fault tree files (alternate description, alternate text)		X
N.10	Event tree files (description, graphics, logic, attributes, linking rules, recovery rules, partition, primary text)	X	X
N.11	Event tree files (alternate description, alternate Text)		X
N.12	End state files (description, cut sets, textual information, primary text)	X	X
N.13	End state files (alternate description, alternate text)		X
N.14	Sequence files (description, logic, cut sets, attributes, recovery rules, primary text)	X	X
N.15	Sequence files (partitions, alternate description, alternate text)		X
N.16	Gate information files (description, attributes)	X	X
N.17	Gate information files (alternate description)		X
N.18	Change Set files (description, information)	X	X
N.19	Change Set files (attributes, alternate description)		X
N.20	Histogram files (description, information)	X	X
N.21	Histogram files (attributes, alternate description)		X
N.22	Slice files (description, basic events, information)	X	X
N.23	Slice files (attributes, alternate description)		X
O	**Model Creation Logic Gate Types** (Max 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	Left/right transfer marker	X	X
O.8	Undeveloped transfer	X	X
O.9	Inhibit gate	X	X
O.10	Basic event	X	X
O.11	Boxed basic event	X	X
O.12	Undeveloped basic event	X	X
O.13	Table of basic events	X	X
O.14	House event	X	X
O.15	Vertical/horizontal text box	X	X
P	**Uncertainty Calculations Monte Carlo and Latin Hyper Cube Sampling**		
P.1	Normal distribution	X	X
P.2	Lognormal distribution	X	X
P.3	Beta distribution	X	X
P.4	Chi Squared distribution	X	X
P.5	Exponential distribution	X	X

Item	Feature Description	Version 6.x	Version 7.x
P.6	Uniform distribution	X	X
P.7	Constrained non-informative distribution	X	X
P.8	Gamma distribution	X	X
P.9	Maximum entropy distribution	X	X
P.10	Dirichlet distribution	X	X
P.11	Seismic log normal analysis	X	X
P.12	Histogram distribution	X	X
P.13	Triangular distribution		X
Q	**Uncertainty Calculations Parameter Settings**		
Q.1	User defined seed, sample size, number of iterations	X	X
Q.2	Output intermediate values to file	X	X
Q.3	Output intermediate values in CSV format		X
R	**General Support Features**		
R.1	Sensitivity wizard		X
R.2	Importance measures wizard		X
R.3	Embedded macro capability		X
R.4	Editing user information		X
R.5	Page numbering control on graphic format	X	X
R.6	Conversion from alpha to graphic format	X	X
R.7	On-line context sensitive help		X
R.8	Parallel processing (Linux only)		X
R.9	Database recovery	X	X
R.10	Designate output folder location		X
R.11	Graphical export to windows metafiles	X	X
S	**General Support Features Report Generation**		
S.1	Project reports (summary, text, letter report, statistics, custom report)	X	X
S.2	Project reports (fault tree recovery rules, sequence recovery rules, partition rules, uncertainty reports)		X
S.3	Attributes (system, location, failure mode, basic event, train type, custom reports)	X	X
S.4	Basic event (overview, probability, uncertainty, seismic, transformation, cross reference, custom reports)	X	X
S.5	Basic event (compound event, developed events, template events, text information)		X
S.6	Fault tree (summary, logic, graphics, cut sets, importance measures, cross reference, custom reports)	X	X
S.7	Fault tree (recovery rules, text information)		X
S.8	Event tree (logic, graphics, initiating events, cross reference, custom reports)	X	X
S. 9	Event tree (linkage rules, recovery rules, partition rules, text information)		X
S.10	End state (summary, cut sets, importance measures, cross reference, custom reports)	X	X
S.11	End state (text information)		X
S.12	Sequence (summary, logic, cut sets, importance measures, custom reports)	X	X
S.13	Sequence (recovery rules, partition rules, text information)		X

Item	Feature Description	Version 6.x	Version 7.x
S.14	Change Set (summary, class, single, text information, custom reports)		X
S.15	Flag Set (summary, flag set events, cross reference, text information, custom reports)		X
S.16	Gate (cross reference, custom reports)	X	X
S.17	Histogram (summary, detailed, custom report)	X	X
S.18	Histogram (cross reference)		X
S.19	Slice (summary, rule summary, slice events, slice rule, custom reports)		X
S.20	User Info/preferences	X	X
S.21	SPAR model report outputs		X
T	**Report Format Types:**		
T.1	ASCII	X	X
T.2	RTF		X
T.3	HTML		X
U	**General Analysis Types**		
U.1	Initiating Event Analysis (formerly GEM)	X	X
U.2	Condition Assessment Analysis (formerly GEM)	X	X
U.3	Accident Sequence Precursor	X	X
U.4	User Define Analysis types	X	X
V	**Application Program Interface**		
V.1	Microsoft © Visual Basic Interface	X	X
V.2	Microsoft © Visual C\C++ Interface	X	X
V.3	Borland © Delphi Object Oriented Pascal		X

APPENDIX B

SAPHIRE QA Process Checklist and Change Forms

APPENDIX B – SAPHIRE QA Process Checklist and Change Forms

1. 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		

2. The development team obtains and retains change request information.

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

4. Requirements derived from NRC Forms 173 and 189 are verified and validated for implementation into automated test scripts.

OK		**Comments:**
Discrepancy		
N/A		

5. NRC Forms 173 and 189 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		

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

OK		**Comments:**
Discrepancy		
N/A		

7. 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		

8. 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		

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

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

11. Test scripts simulate typical user input.

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

13. Test results are saved and compared against expected results.

OK		**Comments:**
Discrepancy		
N/A		

14. User documentation is updated upon completion of each new release.

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

17. Only authorized changes are made to the software release.

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

19. 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		

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

OK		**Comments:**
Discrepancy		
N/A		

21. 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		

22. 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		

23. Bugs are categorized by severity.

OK		**Comments:**
Discrepancy		
N/A		

24. Change requests and bug fixes are placed under configuration control.

OK		**Comments:**
Discrepancy		
N/A		

25. Version control software tracks changes by author and time.

OK		**Comments:**
Discrepancy		
N/A		

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

OK		**Comments:**
Discrepancy		
N/A		

27. 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		

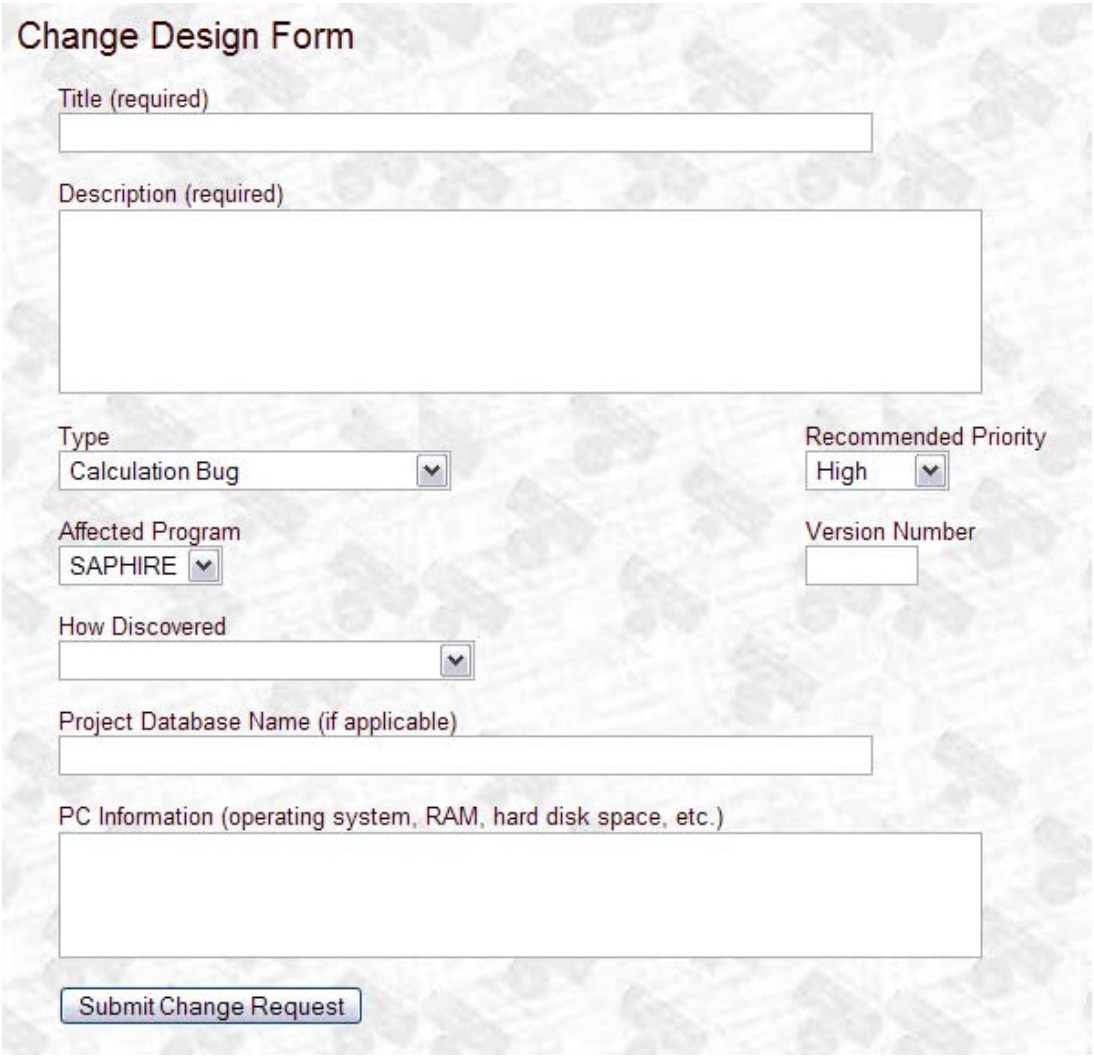
28. Test scripts are reviewed to ensure that requirements are tested adequately, completely, and correctly. (Good Business Practice) (Sample)

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 (as of November, 2003) 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 Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
BYRN-01	Solve Fault Trees	PASSED	TEST-01	[PBYRN-01]
BYRN-02	Core Damage Frequency	PASSED		[PBYRN-02]
BYRN-03	Condition AFW out of service for 72 hours	PASSED	TEST-03	[PBYRN-03]
BYRN-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PBYRN-04]
BYRN-05	Transient - No other failures	PASSED	TEST-05	[PBYRN-05]
BYRN-06	Small LOCA - No other failures	PASSED	TEST-06	[PBYRN-06]
BYRN-07	SGTR - no other failures	PASSED	TEST-07	[PBYRN-07]
BYRN-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[PBYRN-08]
BYRN-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PBYRN-09]
BYRN-10	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PBYRN-10]
BYRN-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PBYRN-11]
BYRN-12	Transient - AFW failed	PASSED	TEST-12	[PBYRN-12]
PBOT-01	Solve Fault Trees	PASSED	TEST-01	[PPBOT-01]
PBOT-02	Core Damage Frequency	PASSED	TEST-02	[PPBOT-02]
PBOT-03	Condition HPCI out of service for 72 hours	PASSED	TEST-03	[PPBOT-03]
PBOT-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PPBOT-04]
PBOT-05	Transient - No other failures	PASSED	TEST-05	[PPBOT-05]
PBOT-06	Small LOCA - No other failures	PASSED	TEST-06	[PPBOT-06]
PBOT-07	Grid-related LOOP - no other failures	PASSED	TEST-08	[PPBOT-07]
PBOT-08	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PPBOT-08]
PBOT-09	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PPBOT-09]
PBOT-10	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PPBOT-10]
PBOT-11	Transient - HPCI failed	PASSED	TEST-12	[PPBOT-11]
DRES-01	Solve Fault Trees	PASSED	TEST-01	[PDRES-01]
DRES-02	Core Damage Frequency	PASSED	TEST-02	[PDRES-02]
DRES-03	Condition HPCI out of service for 72	PASSED	TEST-03	[PDRES-03]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
	hours			
DRES-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PDRES-04]
DRES-05	Transient - No other failures	PASSED	TEST-05	[PDRES-05]
DRES-06	Small LOCA - No other failures	PASSED	TEST-06	[PDRES-06]
DRES-07	Grid-related LOOP - no other failures	PASSED	TEST-08	[PDRES-07]
DRES-08	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PDRES-08]
DRES-09	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PDRES-09]
DRES-10	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PDRES-10]
DRES-11	Transient - HPCI failed	PASSED	TEST-12	[PDRES-11]
GGUL-01	Solve Fault Trees	PASSED	TEST-01	[PGGUL-01]
GGUL-02	Core Damage Frequency	PASSED	TEST-02	[PGGUL-02]
GGUL-03	Condition HPCI out of service for 72 hrs	PASSED	TEST-03	[PGGUL-03]
GGUL-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PGGUL-04]
GGUL-05	Transient - No other failures	PASSED	TEST-05	[PGGUL-05]
GGUL-06	Small LOCA - No other failures	PASSED	TEST-06	[PGGUL-06]
GGUL-07	Grid-related LOOP - no other failures	PASSED	TEST-08	[PGGUL-07]
GGUL-08	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PGGUL-08]
GGUL-09	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PGGUL-09]
GGUL-10	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PGGUL-10]
GGUL-11	Transient - HPCI failed	PASSED	TEST-12	[PGGUL-11]
MIL3-01	Solve Fault Trees	PASSED	TEST-01	[PMIL3-01]
MIL3-02	Core Damage Frequency	PASSED	TEST-02	[PMIL3-02]
MIL3-03	Condition AFW out of service for 72 hours	PASSED	TEST-03	[PMIL3-03]
MIL3-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PMIL3-04]
MIL3-05	Transient - No other failures	PASSED	TEST-05	[PMIL3-05]
MIL3-06	Small LOCA - No other failures	PASSED	TEST-06	[PMIL3-06]
MIL3-07	SGTR - no other failures	PASSED	TEST-07	[PMIL3-07]
MIL3-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[PMIL3-08]
MIL3-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PMIL3-09]
MIL3-10	Severe Weather LOOP - no other	PASSED	TEST-10	[PMIL3-10]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
	failures			
MIL3-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PMIL3-11]
MIL3-12	Transient - AFW failed	PASSED	TEST-12	[PMIL3-12]
OCON-01	Solve Fault Trees	PASSED	TEST-01	[POCON-01]
OCON-02	Core Damage Frequency	PASSED	TEST-02	[POCON-02]
OCON-03	Condition EFW out of service for 72 hours	PASSED	TEST-03	[POCON-03]
OCON-04	Condition 3TC out of service for 3 months	PASSED	TEST-04	[POCON-04]
OCON-05	Transient - No other failures	PASSED	TEST-05	[POCON-05]
OCON-06	Small LOCA - No other failures	PASSED	TEST-06	[POCON-06]
OCON-07	SGTR - no other failures	PASSED	TEST-07	[POCON-07]
OCON-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[POCON-08]
OCON-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[POCON-09]
OCON-10	Severe Weather LOOP - no other failures	PASSED	TEST-10	[POCON-10]
OCON-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[POCON-11]
OCON-12	Transient - EFW failed	PASSED	TEST-12	[POCON-12]
OYST-01	Solve Fault Trees	PASSED	TEST-01	[POYST-01]
OYST-02	Core Damage Frequency	PASSED	TEST-02	[POYST-02]
OYST-03	Condition MFW out of service for 72 hours	PASSED	TEST-03	[POYST-03]
OYST-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[POYST-04]
OYST-05	Transient - No other failures	PASSED	TEST-05	[POYST-05]
OYST-06	Small LOCA - No other failures	PASSED	TEST-06	[POYST-06]
OYST-07	Grid-related LOOP - no other failures	PASSED	TEST-08	[POYST-07]
OYST-08	Plant-centered LOOP - no other failures	PASSED	TEST-09	[POYST-08]
OYST-09	Severe Weather LOOP - no other failures	PASSED	TEST-10	[POYST-09]
OYST-10	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[POYST-10]
OYST-11	Transient - MFW failed	PASSED	TEST-12	[POYST-11]
SONG-01	Solve Fault Trees	PASSED	TEST-01	[PSONG-01]
SONG-02	Core Damage Frequency	PASSED	TEST-02	[PSONG-02]
SONG-03	Condition AFW out of service for 72 hours	PASSED	TEST-03	[PSONG-03]
SONG-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PSONG-04]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
SONG-05	Transient - No other failures	PASSED	TEST-05	[PSONG-05]
SONG-06	Small LOCA - No other failures	PASSED	TEST-06	[PSONG-06]
SONG-07	SGTR - no other failures	PASSED	TEST-07	[PSONG-07]
SONG-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[PSONG-08]
SONG-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PSONG-09]
SONG-10	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PSONG-10]
SONG-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PSONG-11]
SONG-12	Transient - AFW failed	PASSED	TEST-12	[PSONG-12]
STL1-01	Solve Fault Trees	PASSED	TEST-01	[PSTL1-01]
STL1-02	Core Damage Frequency	PASSED	TEST-02	[PSTL1-02]
STL1-03	Condition AFW out of service for 72 hours	PASSED	TEST-03	[PSTL1-03]
STL1-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PSTL1-04]
STL1-05	Transient - No other failures	PASSED	TEST-05	[PSTL1-05]
STL1-06	Small LOCA - No other failures	PASSED	TEST-06	[PSTL1-06]
STL1-07	SGTR - no other failures	PASSED	TEST-07	[PSTL1-07]
STL1-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[PSTL1-08]
STL1-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PSTL1-09]
STL1-10	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PSTL1-10]
STL1-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PSTL1-11]
STL1-12	Transient - AFW failed	PASSED	TEST-12	[PSTL1-12]
SURY-01	Solve Fault Trees	PASSED	TEST-01	[PSURY-01]
SURY-02	Core Damage Frequency	PASSED	TEST-02	[PSURY-02]
SURY-03	Condition AFW out of service for 72 hours	PASSED	TEST-03	[PSURY-03]
SURY-04	Condition EDG out of service for 3 months	PASSED	TEST-04	[PSURY-04]
SURY-05	Transient - No other failures	PASSED	TEST-05	[PSURY-05]
SURY-06	Small LOCA - No other failures	PASSED	TEST-06	[PSURY-06]
SURY-07	SGTR - no other failures	PASSED	TEST-07	[PSURY-07]
SURY-08	Grid-related LOOP - no other failures	PASSED	TEST-08	[PSURY-08]
SURY-09	Plant-centered LOOP - no other failures	PASSED	TEST-09	[PSURY-09]
SURY-10	Severe Weather LOOP - no other failures	PASSED	TEST-10	[PSURY-10]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
SURY-11	Extreme Severe Weather LOOP - no other failures	PASSED	TEST-11	[PSURY-11]
SURY-12	Transient - AFW failed	PASSED	TEST-12	[PSURY-12]
SUR40-01	Solve Sequence Cutsets	PASSED	TEST-02	[PSUR40-01]
SUR40-02	Project Uncertainty - Monte Carlo Method	PASSED	TEST-13	[PSUR40-02]
TstU-01	Log Normal Distribution using MCS	PASSED	TEST-14	[PTstU-01]
TstU-02	Normal Distribution using MCS	PASSED	TEST-15	[PTstU-02]
TstU-03	Beta Distribution using MCS	PASSED	TEST-16	[PTstU-03]
TstU-04	Chi-Squared Distribution using MCS	PASSED	TEST-17	[PTstU-04]
TstU-05	Exponential Distribution using MCS	PASSED	TEST-18	[PTstU-05]
TstU-06	Uniform Distribution using MCS	PASSED	TEST-19	[PTstU-06]
TstU-07	Gamma Distribution using MCS	PASSED	TEST-20	[PTstU-07]
TstU-08	Maximum Entropy Distribution using MCS	PASSED	TEST-21	[PTstU-08]
TstU-09	Constrained Noninformative Distribution using MCS	PASSED	TEST-24	[PTstU-09]
TstU-10	Seismic Log Normal Distribution using MCS	PASSED	TEST-23	[PTstU-10]
TstU-11	Histogram Distribution using MCS	PASSED	TEST-36	[PTstU-11]
TstU-12	Log Normal Distribution using LHS	PASSED	TEST-25	[PTstU-12]
TstU-13	Normal Distribution using LHS	PASSED	TEST-26	[PTstU-13]
TstU-14	Beta Distribution using LHS	PASSED	TEST-27	[PTstU-14]
TstU-15	Chi-Squared Distribution using LHS	PASSED	TEST-28	[PTstU-15]
TstU-16	Exponential Distribution using LHS	PASSED	TEST-29	[PTstU-16]
TstU-17	Uniform Distribution using LHS	PASSED	TEST-30	[PTstU-17]
TstU-18	Gamma Distribution using LHS	PASSED	TEST-31	[PTstU-18]
TstU-19	Maximum Entropy Distribution using LHS	PASSED	TEST-32	[PTstU-19]
TstU-20	Constrained Noninformative Distribution using LHS	PASSED	TEST-35	[PTstU-20]
TstU-21	Seismic Log Normal Distribution using LHS	PASSED	TEST-34	[PTstU-21]
TstU-22	Histogram Distribution using LHS	PASSED	TEST-37	[PTstU-22]
TstU-23	Sq Constrained Noninformative Distribution using MCS	PASSED	TEST-24	[PTstU-23]
TstU-24	Sq Dirichlet Distribution using MCS	PASSED	TEST-22	[PTstU-24]
BV2-5-01	Gather End States	PASSED	TEST-38	[PBV2-5-01]
BV2-5-02	End State Uncertainty using MCS	PASSED	TEST-39	[PBV2-5-02]
BV2-5-03	End State Uncertainty using LHS	PASSED	TEST-40	[PBV2-5-03]
BV2-5-10	End State Group Uncertainty using MCS	PASSED	TEST-39	[PBV2-5-10]
BV2-5-11	End State Group Uncertainty using LHS	PASSED	TEST-40	[PBV2-5-11]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
SURRY-50-01	Check Sequence Cut Sets without Flag Sets	PASSED	TEST-41	[PSURRY-50-01]
SURRY-50-02	Check Sequence Cut Sets with Flag Sets	PASSED	TEST-41	[PSURRY-50-02]
SURRY-50-03	Check Fault Tree Cut Sets (no flag sets in this db)	PASSED	TEST-41	[PSURRY-50-03]
SURRY-50-04	Check Fault Tree Cut Sets without Flag Sets	PASSED	TEST-41	[PSURRY-50-04]
SURRY-50-05	Check End State Cut Sets	PASSED	TEST-41	[PSURRY-50-05]
SURRY-5O-06	Class Change - All Events	PASSED	TEST-51	[PSURRY-5O-06]
SURRY-5O-07	Class Change - LPR-MOV-* Events	PASSED	TEST-51	[PSURRY-5O-07]
SURRY-5O-08	Single Change - 1 Event	PASSED	TEST-51	[PSURRY-5O-08]
SURRY-5O-09	Marked Change Sets	PASSED	TEST-52	[PSURRY-5O-09]
COM-PEAK-01	Check Sequence Cut Sets without Flag Sets	PASSED	TEST-41	[PCOM-PEAK-01]
COM-PEAK-02	Check Sequence Cut Sets with Flag Sets	PASSED	TEST-41	[PCOM-PEAK-02]
COM-PEAK-03	Check Fault Tree Cut Sets	PASSED	TEST-41	[PCOM-PEAK-03]
COM-PEAK-04	Check Fault Tree Cut Sets without Flag Sets	PASSED	TEST-41	[PCOM-PEAK-04]
COM-PEAK-05	Check End State Cut Sets	PASSED	TEST-41	[PCOM-PEAK-05]
S_LERF-01	Link Level 1 Event Trees	PASSED	TEST-42	[PS_LERF-01]
S_LERF-02	Partition Sequence Cut Sets	PASSED	TEST-43	[PS_LERF-02]
S_LERF-03	Link PDS Trees	PASSED	TEST-44	[PS_LERF-03]
DEMO-01	Fault Tree Fussell-Vesely Importance	PASSED	TEST-45	[PDEMO-01]
DEMO-02	Fault Tree Birnbaum Importance	PASSED	TEST-45	[PDEMO-02]
DEMO-03	Fault Tree Uncertainty Importance	PASSED	TEST-45	[PDEMO-03]
DEMO-04	Sequence Fussell-Vesely Importance	PASSED	TEST-46	[PDEMO-04]
DEMO-05	Sequence Birnbaum Importance	PASSED	TEST-46	[PDEMO-05]
DEMO-06	Sequence Uncertainty Importance	PASSED	TEST-46	[PDEMO-06]

Test Number	Test Description	Pass or Fail Status	Test Reference Number	Test Case Name
DEMO-07	Sequence Fussell-Vesely Group Importance	PASSED	TEST-46	[PDEMO-07]
DEMO-08	Sequence Birnbaum Group Importance	PASSED	TEST-46	[PDEMO-08]
DEMO-09	Sequence Uncertainty Group Importance	PASSED	TEST-46	[PDEMO-09]
DEMO-10	Class Change - All Events	PASSED	TEST-51	[PDEMO-10]
DEMO-11	Class Change - ?-MOV-1 Events	PASSED	TEST-51	[PDEMO-11]
DEMO-12	Single Change - 1 Event	PASSED	TEST-51	[PDEMO-12]
DEMO-13	Marked Change Sets	PASSED	TEST-52	[PDEMO-13]
BV2-5-04	End State Fussell-Vesely Importance	PASSED	TEST-48	[PBV2-5-04]
BV2-5-05	End State Birnbaum Importance	PASSED	TEST-48	[PBV2-5-05]
BV2-5-06	End State Uncertainty Importance	PASSED	TEST-48	[PBV2-5-06]
BV2-5-07	End State Fussell-Vesely Group Importance	PASSED	TEST-48	[PBV2-5-07]
BV2-5-08	End State Birnbaum Group Importance	PASSED	TEST-48	[PBV2-5-08]
BV2-5-09	End State Uncertainty Group Importance	PASSED	TEST-48	[PBV2-5-09]
CR3-01	Solve Fault tree	PASSED	TEST-01	[PCR3-01]
CR3-02	Extract,Delete,Load,Solve	PASSED	TEST-53	[PCR3-02]
CR3-03	Auto page, Solve	PASSED	TEST-54	[PCR3-03]
CR3-04	Save cutsets to end state	PASSED	TEST-54	[PCR3-04]
SEQH_3I-01	Check Sequence Cut Sets	PASSED	TEST-41	[PSEQH_3I-01]

Additional details of each test are shown below:

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

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.

Test-03 Condition Assessment - Auxiliary Feed Water (AFW) 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 conditional core damage probability (CCDP); total core damage probability (CDP); total importance; and CCDP, CDP, and importance for each sequence are verified.

Test-04 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 CDP; total importance; and CCDP, CDP, and importance for each sequence are verified.

Test-05 Initiating Event Assessment - Transient with no other failures.
Scenarios exercise the number of sequences; total CCDP; total 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.

Test-06 Initiating Event Assessment – Small Loss of Coolant Accident (SLOCA) with no other failures.
Scenarios exercise the number of sequences; total CCDP; total 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.

Test-07 Initiating Event Assessment – Steam Generator Tube Rupture with no other failures.
Scenarios exercise the number of sequences; total CCDP; total 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.

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 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.

Test-09 Initiating Event Assessment - Plant-Centered LOOP with no other failures
Scenarios exercise the number of sequences; total CCDP; total 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.

Test-10 Initiating Event Assessment - Severe Weather LOOP with no other failures
Scenarios exercise the number of sequences; total CCDP; total 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.

Test-11 Initiating Event Assessment – Extreme Severe Weather LOOP with no other failures
Scenarios exercise the number of sequences; total CCDP; total 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.

Test-12 Initiating Event Assessment - Transient with AFW Failed
Scenarios exercise the number of sequences; total CCDP; total 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.

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.

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.

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.

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.

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.

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.

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.

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.

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, 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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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.

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, 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Test 55 – Event Tree Linking (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.

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.

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.

Test –58 Base Case Update
All models have fault tree results and accident sequences cut sets converted to the base case. A scenario for fault tree cut sets converted to the base case for comparison to the current case using change sets.

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.

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.

Test– 61 Uncertainty Distributions
All of the uncertainty distribution types are tested.

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.

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.

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.

Test – 65 Event Transformations
This test checks the capability of SAPHIRE to turn one or more basic events into other basic events during the cut set generation process. This feature is primarily used for external events models.

Test - 66 Wrong Results
This test verifies the output of results. The output from the test is compared to incorrect results to verify that the comparison is worked correctly. A LOSP scenario is executed to obtain results for comparison.

Below, the physical output from two of the individual tests, the PBYRN-01 test (solving for fault tree minimal cut sets) and the PBYRN-02 test (solving event trees for core damage cut sets) are shown. Not only are each test graded on a pass/fail, but one should note that each entity (e.g., different fault trees, different sequences) is checked and graded on a pass/fail basis. All total, there are over 250 high-level tests, where each test comprises multiple sub-tests on specific portions of the SAPHIRE software.

TEST CASE: SAPHIRE QA Models (CDF_BYRN) DATE & TIME: 8/6/03 6:09:37 PM

Operating System:Microsoft Windows NT

TEST FOR: SAPHIRE Version 7.20

Opened project: bryn_2qa

[PBYRN-01]Scenario: Solve Fault Trees started at 6:10:00 PM

Generated base case data
Fault trees solved
with prob cut off (1.0E-16)
Fault Tree base case updated

FAULT TREE RESULTS:

Compare Min-Cut and No. of Cut Sets:

Fault Tree	Min-Cut	Status	Failure	Base	Status	Count	Status
ACP-ST	5.300E-001	pass		5.300E-01	pass	1	pass
AFW	3.341E-004	pass		3.341E-04	pass	13	pass
AFW-ATWS	2.425E-002	pass		2.425E-02	pass	14	pass
AFW-L	3.341E-004	pass		3.341E-04	pass	13	pass
AFW-SGTR	3.531E-004	pass		3.531E-04	pass	12	pass
BORATION	1.000E-003	pass		1.000E-03	pass	1	pass
COOLDOWN	3.997E-003	pass		3.997E-03	pass	2	pass
DEP-REC	3.500E-003	pass		3.500E-03	pass	1	pass
EP	2.889E-003	pass		2.889E-03	pass	5	pass
F&B	2.244E-002	pass		2.244E-02	pass	91	pass
F&L	2.244E-002	pass		2.244E-02	pass	91	pass
HPI	9.140E-006	pass		9.140E-06	pass	88	pass

HPI-L	9.140E-006	pass		9.140E-06	pass	88	pass
HPR	2.731E-003	pass		2.731E-03	pass	754	pass
HPR-L	2.731E-003	pass		2.731E-03	pass	754	pass
LPR	2.228E-003	pass		2.228E-03	pass	44	pass
MFW-A	2.000E-001	pass		2.000E-01	pass	1	pass
MFW-NT	5.000E-002	pass		5.000E-02	pass	1	pass
MFW-T	7.840E-002	pass		7.840E-02	pass	2	pass
OP-2H	1.200E-001	pass		1.200E-01	pass	1	pass

Compare Mean:

Fault Tree	Mean	Status	Failure
ACP-ST	0.000E+00	pass	
AFW	0.000E+00	pass	
AFW-ATWS	0.000E+00	pass	
AFW-L	0.000E+00	pass	
AFW-SGTR	0.000E+00	pass	
BORATION	0.000E+00	pass	
COOLDOWN	0.000E+00	pass	
DEP-REC	0.000E+00	pass	
EP	0.000E+00	pass	
F&B	0.000E+00	pass	
F&L	0.000E+00	pass	
HPI	0.000E+00	pass	
HPI-L	0.000E+00	pass	
HPR	0.000E+00	pass	
HPR-L	0.000E+00	pass	
LPR	0.000E+00	pass	
MFW-A	0.000E+00	pass	
MFW-NT	0.000E+00	pass	
MFW-T	0.000E+00	pass	
OP-2H	0.000E+00	pass	

Compare Min-Cut and No. of Cut Sets:

Fault Tree	Min-Cut	Status	Failure	Base	Status	Count	Status
OP-6H	3.600E-002	pass		3.600E-02	pass	1	pass
OP-BD	2.000E-002	pass		2.000E-02	pass	1	pass
OP-SL	6.300E-001	pass		6.300E-01	pass	1	pass
PORV	4.000E-002	pass		4.000E-02	pass	1	pass
PORV-1	1.000E+000	pass		1.000E+00	pass	1	pass
PORV-A	2.716E-001	pass		2.716E-01	pass	9	pass
PORV-L	1.600E-001	pass		1.600E-01	pass	1	pass
PORV-RES	2.454E-004	pass		2.454E-04	pass	6	pass
PORV-SBO	3.700E-001	pass		3.700E-01	pass	1	pass
PRVL-RES	2.454E-004	pass		2.454E-04	pass	6	pass
RCS-DEP	3.997E-003	pass		3.997E-03	pass	2	pass

Compare Mean:

Fault Tree	Mean	Status	Failure
OP-6H	0.000E+00	pass	
OP-BD	0.000E+00	pass	
OP-SL	0.000E+00	pass	
PORV	0.000E+00	pass	
PORV-1	0.000E+00	pass	
PORV-A	0.000E+00	pass	
PORV-L	0.000E+00	pass	
PORV-RES	0.000E+00	pass	
PORV-SBO	0.000E+00	pass	
PRVL-RES	0.000E+00	pass	
RCS-DEP	0.000E+00	pass	

Compare Min-Cut and No. of Cut Sets:

Fault Tree	Min-Cut	Status	Failure	Base	Status	Count	Status
RCS-SG	3.738E-002	pass		3.738E-02	pass	3	pass
RCS-SG1	2.766E-002	pass		2.766E-02	pass	2	pass
RCSPRESS	1.303E-002	pass		1.303E-02	pass	2	pass
RHR	3.298E-003	pass		3.298E-03	pass	45	pass
RT	5.529E-006	pass		5.529E-06	pass	3	pass
RT-L	8.900E-008	pass		8.900E-08	pass	1	pass
SEALLOCA	3.500E-002	pass		3.500E-02	pass	1	pass
SG-DEP	1.000E-005	pass		1.000E-05	pass	1	pass
SGCOOL	2.005E-001	pass		2.005E-01	pass	5	pass
SGCOOL-L	3.404E-001	pass		3.404E-01	pass	5	pass
SGISOL	1.099E-002	pass		1.099E-02	pass	2	pass
SGISOL1	1.228E-002	pass		1.228E-02	pass	4	pass
SLOCA-NR	4.300E-001	pass		4.300E-01	pass	1	pass
THROTTLE	1.000E-002	pass		1.000E-02	pass	1	pass

Compare Mean:

Fault Tree	Mean	Status	Failure
RCS-SG	0.000E+00	pass	
RCS-SG1	0.000E+00	pass	
RCSPRESS	0.000E+00	pass	
RHR	0.000E+00	pass	
RT	0.000E+00	pass	
RT-L	0.000E+00	pass	
SEALLOCA	0.000E+00	pass	
SG-DEP	0.000E+00	pass	
SGCOOL	0.000E+00	pass	
SGCOOL-L	0.000E+00	pass	
SGISOL	0.000E+00	pass	
SGISOL1	0.000E+00	pass	
SLOCA-NR	0.000E+00	pass	
THROTTLE	0.000E+00	pass	

Scenario: Solve Fault Trees completed at 6:10:42 PM

[PBYRN-02]Scenario: Core Damage Frequency Test started at 6:10:43 PM
Generated base case data
Sequences solved
with prob cut off (1.0E-16) and with recovery
Event Tree base case updated
SEQUENCE RESULTS:
Compare MinCut and No. of Cut Sets:

Event Tree	Sequence	Min-Cut	Status	Failure	Base	Status	Count	Status
LOOP	05	5.403E-12	pass		5.403E-12	pass	105	pass
LOOP	07	5.303E-14	pass		5.303E-14	pass	43	pass
LOOP	09	1.692E-11	pass		1.692E-11	pass	208	pass
LOOP	10	2.376E-11	pass		2.376E-11	pass	58	pass
LOOP	13	2.395E-12	pass		2.395E-12	pass	441	pass
LOOP	16	1.185E-12	pass		1.185E-12	pass	270	pass
LOOP	17	9.942E-11	pass		9.942E-11	pass	155	pass
LOOP	18-02	4.499E-10	pass		4.499E-10	pass	5	pass
LOOP	18-05	2.877E-13	pass		2.877E-13	pass	48	pass
LOOP	18-07	2.595E-15	pass		2.595E-15	pass	14	pass
LOOP	18-08	5.188E-15	pass		5.188E-15	pass	13	pass
LOOP	18-09	5.140E-10	pass		5.140E-10	pass	5	pass
LOOP	18-11	2.642E-10	pass		2.642E-10	pass	5	pass
LOOP	18-14	1.683E-13	pass		1.683E-13	pass	37	pass
LOOP	18-16	1.005E-15	pass		1.005E-15	pass	6	pass
LOOP	18-17	2.873E-15	pass		2.873E-15	pass	9	pass
LOOP	18-18	3.019E-10	pass		3.019E-10	pass	5	pass
LOOP	18-20	4.354E-10	pass		4.354E-10	pass	10	pass
LOOP	18-22	1.350E-10	pass		1.350E-10	pass	29	pass
LOOP	19	1.424E-12	pass		1.424E-12	pass	1	pass

Compare Mean:

Event Tree	Sequence	Mean	Status	Failure
LOOP	05	0.000E+00	pass	
LOOP	07	0.000E+00	pass	
LOOP	09	0.000E+00	pass	
LOOP	10	0.000E+00	pass	
LOOP	13	0.000E+00	pass	
LOOP	16	0.000E+00	pass	
LOOP	17	0.000E+00	pass	
LOOP	18-02	0.000E+00	pass	
LOOP	18-05	0.000E+00	pass	
LOOP	18-07	0.000E+00	pass	
LOOP	18-08	0.000E+00	pass	
LOOP	18-09	0.000E+00	pass	
LOOP	18-11	0.000E+00	pass	
LOOP	18-14	0.000E+00	pass	
LOOP	18-16	0.000E+00	pass	
LOOP	18-17	0.000E+00	pass	
LOOP	18-18	0.000E+00	pass	
LOOP	18-20	0.000E+00	pass	
LOOP	18-22	0.000E+00	pass	
LOOP	19	0.000E+00	pass	

Compare MinCut and No. of Cut Sets:

Event Tree	Sequence	Min-Cut	Status	Failure	Base	Status	Count	Status
SGTR	03	5.920E-11	pass		5.920E-11	pass	82	pass
SGTR	04	7.172E-11	pass		7.172E-11	pass	4	pass
SGTR	05	1.630E-11	pass		1.630E-11	pass	1	pass
SGTR	08	2.496E-12	pass		2.496E-12	pass	228	pass
SGTR	09	3.031E-12	pass		3.031E-12	pass	24	pass
SGTR	10	6.161E-13	pass		6.161E-13	pass	3	pass
SGTR	11	2.156E-10	pass		2.156E-10	pass	3	pass
SGTR	13	1.363E-13	pass		1.363E-13	pass	48	pass
SGTR	14	0.000E+00	pass		0.000E+00	pass	0	pass

Compare Mean:

Event Tree	Sequence	Mean	Status	Failure
SGTR	03	0.000E+00	pass	
SGTR	04	0.000E+00	pass	
SGTR	05	0.000E+00	pass	
SGTR	08	0.000E+00	pass	
SGTR	09	0.000E+00	pass	
SGTR	10	0.000E+00	pass	
SGTR	11	0.000E+00	pass	
SGTR	13	0.000E+00	pass	
SGTR	14	0.000E+00	pass	

Compare MinCut and No. of Cut Sets:

Event Tree	Sequence	Min-Cut	Status	Failure	Base	Status	Count	Status
SGTR	16	2.860E-15	pass		2.860E-15	pass	10	pass
SGTR	17	0.000E+00	pass		0.000E+00	pass	0	pass
SGTR	18	7.546E-16	pass		7.546E-16	pass	4	pass
SGTR	21	1.312E-14	pass		1.312E-14	pass	28	pass
SGTR	22	6.463E-15	pass		6.463E-15	pass	17	pass
SGTR	23	1.483E-15	pass		1.483E-15	pass	6	pass
SGTR	26	2.884E-16	pass		2.884E-16	pass	3	pass
SGTR	27	8.277E-17	pass		8.277E-17	pass	2	pass
SGTR	28	0.000E+00	pass		0.000E+00	pass	0	pass
SGTR	29	1.975E-14	pass		1.975E-14	pass	21	pass
SGTR	31	2.431E-17	pass		2.431E-17	pass	1	Pass
SGTR	32	0.000E+00	pass		0.000E+00	pass	0	Pass
SGTR	34	0.000E+00	pass		0.000E+00	pass	0	Pass
SGTR	35	0.000E+00	pass		0.000E+00	pass	0	Pass
SGTR	36	0.000E+00	pass		0.000E+00	pass	0	Pass
SGTR	39	6.887E-15	pass		6.887E-15	pass	23	Pass
SGTR	41	4.450E-17	pass		4.450E-17	pass	1	Pass
SGTR	42	8.230E-14	pass		8.230E-14	pass	16	Pass
SGTR	43	1.419E-13	pass		1.419E-13	pass	26	Pass
SGTR	44	9.012E-12	pass		9.012E-12	pass	3	Pass

Compare Mean:

Event Tree	Sequence	Mean	Status	Failure
SGTR	16	0.000E+00	pass	
SGTR	17	0.000E+00	pass	
SGTR	18	0.000E+00	pass	
SGTR	21	0.000E+00	pass	
SGTR	22	0.000E+00	pass	
SGTR	23	0.000E+00	pass	
SGTR	26	0.000E+00	pass	
SGTR	27	0.000E+00	pass	
SGTR	28	0.000E+00	pass	
SGTR	29	0.000E+00	pass	
SGTR	31	0.000E+00	pass	
SGTR	32	0.000E+00	pass	
SGTR	34	0.000E+00	pass	
SGTR	35	0.000E+00	pass	
SGTR	36	0.000E+00	pass	
SGTR	39	0.000E+00	pass	
SGTR	41	0.000E+00	pass	
SGTR	42	0.000E+00	pass	
SGTR	43	0.000E+00	pass	
SGTR	44	0.000E+00	pass	

Compare MinCut and No. of Cut Sets:

Event Tree	Sequence	MinCut	Status	Failure	Base	Status	Count	Status
SLOCA	04	9.088E-10	pass		9.088E-10	pass	357	Pass
SLOCA	06	1.092E-11	pass		1.092E-11	pass	236	Pass
SLOCA	07	7.692E-12	pass		7.692E-12	pass	66	Pass
SLOCA	11	8.798E-14	pass		8.798E-14	pass	62	Pass
SLOCA	13	5.689E-16	pass		5.689E-16	pass	9	Pass
SLOCA	14	2.304E-15	pass		2.304E-15	pass	10	Pass
SLOCA	17	9.983E-15	pass		9.983E-15	pass	30	Pass
SLOCA	19	0.000E+00	pass		0.000E+00	pass	0	Pass
SLOCA	21	4.728E-15	pass		4.728E-15	pass	24	Pass

Compare Mean:

Event Tree	Sequence	Mean	Status	Failure
SLOCA	04	0.000E+00	pass	
SLOCA	06	0.000E+00	pass	
SLOCA	07	0.000E+00	pass	
SLOCA	11	0.000E+00	pass	
SLOCA	13	0.000E+00	pass	
SLOCA	14	0.000E+00	pass	
SLOCA	17	0.000E+00	pass	
SLOCA	19	0.000E+00	pass	
SLOCA	21	0.000E+00	pass	

Compare MinCut and No. of Cut Sets:

Event Tree	Sequence	MinCut	Status	Failure	Base	Status	Count	Status
SLOCA	22	1.920E-13	pass		1.920E-13	pass	26	pass
SLOCA	23	1.288E-11	pass		1.288E-11	pass	3	pass
TRANS	05	3.420E-12	pass		3.420E-12	pass	108	pass
TRANS	07	2.545E-14	pass		2.545E-14	pass	49	pass
TRANS	08	2.362E-13	pass		2.362E-13	pass	44	pass
TRANS	13	8.295E-14	pass		8.295E-14	pass	69	pass
TRANS	15	1.995E-16	pass		1.995E-16	pass	6	pass
TRANS	16	1.493E-14	pass		1.493E-14	pass	14	pass
TRANS	19	9.935E-13	pass		9.935E-13	pass	640	pass
TRANS	20	3.271E-11	pass		3.271E-11	pass	134	pass
TRANS	21-04	3.695E-13	pass		3.695E-13	pass	62	pass
TRANS	21-06	1.817E-15	pass		1.817E-15	pass	9	pass
TRANS	21-07	1.371E-12	pass		1.371E-12	pass	3	pass
TRANS	21-11	7.246E-14	pass		7.246E-14	pass	36	pass
TRANS	21-13	0.000E+00	pass		0.000E+00	pass	0	pass
TRANS	21-14	2.742E-13	pass		2.742E-13	pass	3	pass
TRANS	21-15	6.675E-12	pass		6.675E-12	pass	21	pass
TRANS	21-16	1.788E-11	pass		1.788E-11	pass	6	pass

Compare Mean:

Event Tree	Sequence	Mean	Status	Failure
SLOCA	22	0.000E+00	pass	
SLOCA	23	0.000E+00	pass	
TRANS	05	0.000E+00	pass	
TRANS	07	0.000E+00	pass	
TRANS	08	0.000E+00	pass	
TRANS	13	0.000E+00	pass	
TRANS	15	0.000E+00	pass	
TRANS	16	0.000E+00	pass	
TRANS	19	0.000E+00	pass	
TRANS	20	0.000E+00	pass	
TRANS	21-04	0.000E+00	pass	
TRANS	21-06	0.000E+00	pass	
TRANS	21-07	0.000E+00	pass	
TRANS	21-11	0.000E+00	pass	
TRANS	21-13	0.000E+00	pass	
TRANS	21-14	0.000E+00	pass	
TRANS	21-15	0.000E+00	pass	
TRANS	21-16	0.000E+00	pass	

Scenario: Core Damage Frequency Test completed at 6:11:42 PM

```
TEST CASE COMPLETE: at 6:11:45 PM
```

NRC FORM 335 (2-89)
NRCM 1102, 3201. 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

(See Instructions on the reverse)

1. REPORT NUMBER (Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)
NUREG/CR-6952
INL/EXT-05-00655

2. TITLE AND SUBTITLE
Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 6 Quality Assurance Manual

3. DATE REPORT PUBLISHED
MONTH: September
YEAR: 2008

4. FIN OR GRANT NUMBER
N6203

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

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) is a software application developed for performing a complete probabilistic risk assessment using a personal computer running the Microsoft Windows operating system. SAPHIRE is primarily 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, what constitutes its parts, and limitations of those processes.

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

13. AVAILABILITY STATEMENT
Unlimited

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

15. NUMBER OF PAGES

16. PRICE

NRC FORM 335 (2-89)