NUREG/CR-6116
INEL-94/0039
Vol. 9

Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 5.0

Verification and Validation (V&V) Manual

Prepared by
J.L. Jones, M.B. Calley, E.L. Capps, S.L. Zeigler, W.J. Galyean, S.D. Novack, C.L. Smith, L.M. Wolfram

Idaho National Laboratory
Lockheed Idaho Technologies Company

Prepared for
U.S. Nuclear Regulatory Commission

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, 2120 L Street, NW., Lower Level, Washington, DC 20555-0001

2. The Superintendent of Documents, U.S. Government Printing Office, P. O. Box 37082, Washington, DC 20402-9328

3. The National Technical Information Service, Springfield, VA 22161-0002

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 investigation 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 Government Printing Office: formal NRC staff and contractor reports, NRC-sponsored conference proceedings, international agreement reports, grantee reports, 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, Printing and Mail Services Section, U.S. Nuclear Regulatory Commission, Washington, DC 20555-0001.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at the NRC Library, Two White Flint North, 11545 Rockville Pike, Rockville, MD 20852-2738, for use by the public. 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 10018-3308.

DISCLAIMER NOTICE

NUREG/CR-6116
INEL-94/0039
Vol. 9

Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 5.0

Verification and Validation (V&V) Manual

Manuscript Completed: February 1995
Date Published: March 1995

Prepared by
J.L. Jones, M.B. Calley, E.L. Capps, S.L. Zeigler, W.J. Galyean, S.D. Novack, C.L. Smith, L.M. Wolfram

Idaho National Engineering Laboratory
Lockheed Idaho Technologies Company
Idaho Falls, ID 83415

Prepared for
Division of Systems Technology
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
NRC Job Code L2483

ABSTRACT

A verification and validation (V&V) process has been performed for the System Analysis Programs for Hands-on Integrated Reliability Evaluation (SAPHIRE) Version 5.0. SAPHIRE is a set of four computer programs that the Nuclear Regulatory Commission has developed for the performance of probabilistic risk assessments. These programs allow an analyst to perform many of the functions necessary to create, quantify, and evaluate the risk associated with a facility or process being analyzed. The programs included in this set are Integrated Reliability and Risk Analysis System (IRRAS), System Analysis and Risk Assessment (SARA), Models And Results Database (MAR-D), and Fault tree, Event tree, and Piping and instrumentation diagram (FEP) graphical editor.

The intent of this program is to perform a V&V of successive versions of SAPHIRE. Previous efforts have been the V&V of SAPHIRE Version 4.0. The SAPHIRE 5.0 V&V plan is based on the SAPHIRE 4.0 V&V plan with revisions to incorporate lessons learned from the previous effort. Also, the SAPHIRE 5.0 vital and nonvital test procedures are based on the test procedures from SAPHIRE 4.0 with revisions to include the new SAPHIRE 5.0 features as well as to incorporate lessons learned from the previous effort. The majority of the results from the testing was acceptable; however, some discrepancies between expected code operation and actual code operation were identified. Modifications that have been made to SAPHIRE are identified.

Documents in NUREG/CR-6116 Report, Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 5.0

Volume 1 - Technical Reference Manual

Volume 2 - Integrated Reliability and Risk Analysis System (IRRAS) Reference Manual

Volume 3 - Integrated Reliability and Risk Analysis System (IRRAS) Tutorial Manual

Volume 4 - Systems Analysis and Risk Assessment (SARA) System Reference Manual

Volume 5 - Systems Analysis and Risk Assessment (SARA) System Tutorial Manual

Volume 6 - Graphical Evaluation Module (GEM) Reference Manual

Volume 7 - Fault Tree, Event Tree, and Piping & Instrumentation Diagram (FEP) Editors Reference Manual

Volume 8 - Models And Results Database (MAR-D) Reference Manual

Volume 9 - Verification and Validation (V&V)

Volume 10 - Data Loading Manual

Previous Reports in the Series

T. W. Bolander et al., *Verification and Validation of the SAPHIRE Version 4.0 PRA Software Package*, NUREG/CR-6145, EGG-2713, February 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, June 1990.

K. D. Russell, D. M. Snider, M. B. Sattison, H. D. Stewart, D. 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.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The System Analysis Programs for Hands-on Integrated Reliability Evaluation (SAPHIRE) is a set of four computer programs that the Nuclear Regulatory Commission has developed for the performance of probabilistic risk assessments (PRAs). These programs allow an analyst to perform many of the functions necessary to create, quantify, and evaluate the risk associated with a facility or process being analyzed. The programs included in this set are Integrated Reliability and Risk Analysis System (IRRAS), System Analysis and Risk Assessment (SARA), Models And Results Database (MAR-D), and Fault tree, Event tree, and Piping and instrumentation diagram (FEP) graphical editor.

The intent of the this program is verification and validation (V&V) of successive versions of SAPHIRE. Previously, a V&V of SAPHIRE 4.0 was performed. This report documents the V&V of SAPHIRE 5.0.

The primary objective of this project was to determine if the results produced by SAPHIRE are correct. A co-objective was to complete the necessary documentation of the V&V per accepted standards.

The V&V of SAPHIRE 5.0 consisted of the following steps:

1. Preparation of a V&V plan
2. Evaluation of the code development control procedures
3. Test case development
4. V&V testing
5. Documentation of test results and recommendations.

Overall, the above steps are consistent with Institute of Electrical and Electronic Engineers (IEEE) "Standard for Software Verification and Validation Plans" (IEEE 1012-1986) except that there are two steps recommended in IEEE-1012-1986 that were not performed. The two steps not performed were an evaluation of the user documentation and an evaluation of the software requirements specification. The user documentation was not explicitly evaluated, but any discrepancies identified during testing between the user documentation and actual code operation were noted. Originally, it was planned to perform an evaluation of the software requirements specification; however, because a requirements document was not prepared for SAPHIRE 5.0, it was not possible to perform this evaluation.

From the evaluation of the code development control procedures it was found that improvements could be made in the areas of establishing the software level, software requirements documentation, requirements and design reviews, coordination of the development and V&V projects so that there is a more comprehensive test program and that acceptance testing takes place before

a new version of the code is released to the Energy Science and Technology Software Center, and reconsideration of the types of projects where beta test versions are used.

SAPHIRE features were evaluated and either designated as vital or nonvital features. Vital features are those that (a) affect the results of a PRA (core damage frequency, top contributors, etc.) and (b) are essential for completing a PRA analysis such as fault tree analysis, event tree analysis, uncertainty analysis, etc. Nonvital features are those whose malfunction or lack of function will not preclude obtaining the final correct results from SAPHIRE. The nonvital features are graphical fault tree construction, graphical event tree construction, database manipulation, etc. Lack of a nonvital feature can add significant time to a PRA because the work that would have been performed by that feature will have to be done manually or through an alternate method.

The SAPHIRE 5.0 vital and nonvital features test procedures are based on those from the SAPHIRE 4.0 V&V effort, but have been revised to include the new SAPHIRE 5.0 features as well as to incorporate lessons learned from the previous effort. Additionally, user interviews were conducted to identify problems and difficulties that users of prerelease versions of SAPHIRE 5.0 have encountered. As necessary, tests were added to the V&V test procedures based on these user-identified problems and difficulties, thus helping to ensure that the test procedures are complete.

For SAPHIRE 5.0, 73 vital features were identified for testing. However, it was recognized that performing all 73 tests would not be possible within the time available. Vital features were ranked as high, medium, or low. Only those vital features ranked as high were tested. The final testing covered 38 vital features.

Software test records were prepared for all vital features tested. Anomaly reports were prepared for all discrepancies identified between the way SAPHIRE should operate and actual operation. Discrepancies were categorized as either critical or noncritical. A discrepancy is categorized as critical if the results are incorrect or the potential exists for the results to be misleading. A discrepancy is categorized as noncritical if the results are correct but the option is difficult to use.

The majority of the results were determined to be acceptable. However, critical discrepancies were identified for some vital features. The critical discrepancies occurred in the four areas of cut set generation function, event probability cutoff feature, uncertainty analysis, and database conversion routine.

Software test records were not prepared for the nonvital features testing. Instead, a test specification and a test procedure were developed, and items were checked off the test procedure as they were completed. All discrepancies identified with the nonvital features were documented on anomaly reports. For the nonvital features, no critical discrepancies were identified.

The discrepancies were provided to the code developers. With the exception of the discrepancy associated with the event probability cutoff feature, modifications were made to SAPHIRE to correct the identified discrepancies. The event probability cutoff feature discrepancy will be corrected with a later release of SAPHIRE. The modifications made to SAPHIRE were not retested.

FOREWORD

The U.S. Nuclear Regulatory Commission (NRC) has developed a powerful suite of personal computer programs for the performance of probabilistic risk assessments (PRAs). This suite of programs, known as the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE), allows an analyst to perform many of the functions necessary to create, quantify, and evaluate the risk associated with a facility or process being analyzed.

These programs include software to define the database structure; to create, analyze, and quantify the data; and to display results and perform sensitivity analyses. The programs included in this suite are as follows: Models And Results Database (MAR-D) software, Integrated Reliability and Risk Analysis System (IRRAS) software, System Analysis and Risk Assessment (SARA) software, and Fault tree, Event tree, and P&ID (FEP) graphical editor software. Each of these programs performs a specific function in taking a PRA from the conceptual state all the way to publication.

MAR-D is a program that is used primarily for PRA data loading. This program defines a common relational database structure that is used by the entire suite of programs. This structure allows all of the software to access and manipulate data created by other software in the system without performing a lengthy conversion. Therefore, data created by IRRAS is immediately available to SARA for sensitivity analysis. The MAR-D program also provides the facilities for loading and unloading of PRA data from the relational database structure used to store the data. A simple ASCII data format is used for interchange with other PRA software not included in NRC's suite of programs. This feature allows for compatibility with previously developed software systems and allows for maximum data interchange. Elements of this software are included with both IRRAS and SARA to allow these programs to load and unload data in the MAR-D format. Normally, the entire MAR-D software is used only by those performing a data loading function and is not required by the end user. Documentation for MAR-D Version 5.0 is available as NUREG/CR-6116, Volume 8. It should be noted that whenever the MAR-D database structure is changed, it necessitates changes in the remaining codes (i.e., IRRAS, SARA, and FEP). Therefore, the code version numbers are changed in unison. Each version set must be used together to maintain compatibility.

IRRAS is a program developed for the purpose of performing those functions necessary to create and analyze a complete PRA. This program includes functions to allow the user to create event trees and fault trees, to define accident sequences and basic event failure data, to solve system and accident sequence fault trees, to quantify cut sets, and to perform uncertainty analysis on the results. Also included in this program are features to allow the analyst to generate reports and displays that can be used to document the results of an analysis. Because this software is a very detailed technical tool, the user of this program should be familiar with PRA concepts and the methods used to perform these analyses. Although IRRAS has been designed to be user friendly and makes the process of performing a PRA easier, the complexity of this type of analysis requires a user with a more detailed understanding of PRA concepts than is required by other tools in this suite. The IRRAS 5.0 reference manual and tutorial are available as NUREG/CR-6116, Volumes 2 and 3, respectively. In addition, a technical document that provides information on the principles and algorithms used in the construction and operation of IRRAS and SARA is available as NUREG/CR-6116, Volume 1.

SARA is a program that allows the user to review the results of a PRA and to perform limited sensitivity analyses on these results. It is limited primarily to the extent that changes in the plant model can be accommodated by using the cut set editor. If other than simple changes are being simulated, then IRRAS should be used so that new cut sets can be accurately generated. This tool is intended to be used by a less technically-oriented user and does not require the level of understanding of PRA concepts required by IRRAS. With this program a user can review the information generated by a PRA analyst and compare the results to those generated by making limited modifications to the data in the PRA. Also included in this program is the ability to graphically display the information stored in the MAR-D database. This information includes event trees, fault trees, P&IDs, and uncertainty distributions. The user of this program can gain a better understanding of the results of a PRA without getting into the details of the construction and analysis work behind the PRA. The SARA reference manual and tutorial are available as NUREG/CR-6116, Volumes 4 and 5, respectively.

FEP is a program developed to provide a common access to the suite of graphical tools developed for performing risk assessment. These tools include the graphical fault tree, event tree, and P&ID editors. The fault tree and event tree editors are available through IRRAS; however, the P&ID editor is only accessible through FEP. The fault tree editor allows the user to construct and modify graphical fault trees. The event tree editor allows the analyst to construct and modify graphical event trees. The P&ID editor allows the user to construct and modify plant drawings. These drawings can then be used to document the modeling used in a PRA. These editors are an integral part of a PRA. With the FEP tool, the user need not be concerned with the complexity of the IRRAS program if the need is only to generate one of these graphical displays. The FEP Reference Manual is available as NUREG/CR-6116, Volume 7.

ACKNOWLEDGMENTS

The authors would like to express their appreciation to Scott T. Beck for his evaluation of the seismic analysis feature.

SAPHIRE Version 5.0
Volume 9—Verification and Validation

1. INTRODUCTION

The System Analysis Programs for Hands-on Integrated Reliability Evaluation (SAPHIRE) is used in a variety of regulatory applications. SAPHIRE was developed by the U.S. Nuclear Regulatory Commission (NRC) to support applications in the areas of:

- Analyzing the risk implications of plant designs, systems operations, and procedures
- Assessing the effectiveness of existing and proposed regulations, including backfits
- Evaluating the significance of operational occurrences
- Prioritizing generic safety issues, research and licensing programs, and inspection activities
- Assisting the Committee to Review Generic Requirements (CRGR) in tracking the progress that required plant modifications made toward improved safety levels.

Examples of these areas include the resolution of generic safety issues (GSIs), examining risk-based inspection strategies, analyzing a number of multi-plant action (MPA) items, and performing a sensitivity study on the significance of motor-operated valve (MOV) failure rates. Because of the extensive use of SAPHIRE in regulatory applications, it was determined that the need existed for verification and validation (V&V) of the operation of the code to ensure that it is adequate to perform accident frequency and risk calculations.

The primary objective of this project was to determine via a structured and systematic V&V process if the results produced by SAPHIRE are correct, and if not, to identify those areas requiring correction. In doing this, the V&V process would be documented per accepted industry standards.

This report documents the V&V of SAPHIRE Version 5.0 dated July 20, 1994. It is the intent of the NRC to validate and verify subsequent versions of SAPHIRE. Previous efforts have been the V&V of SAPHIRE Version 4.0.[1] Lessons learned from that effort have been incorporated into the SAPHIRE 5.0 V&V, and the lessons learned from this effort will be incorporated into the V&V of subsequent SAPHIRE versions.

2. APPROACH

The V&V of SAPHIRE consisted of the following steps:

1. Preparation of a V&V plan
2. Evaluation of the code development control procedures
3. Test case development
4. V&V testing
5. Documentation of test results and recommendations.

The above steps are consistent with IEEE "Standard for Software Verification and Validation Plans" (IEEE 1012-1986),[2] except that there are two steps recommended in IEEE-1012-1986 that were not performed. The two steps not performed were an evaluation of the user documentation and an evaluation of the software requirements specification. The user documentation was not explicitly evaluated, but any discrepancies identified during testing between the user documentation and actual code operation were noted. Originally, it was planned to perform an evaluation of the software requirements specification; however, because a requirements document was not prepared for SAPHIRE 5.0, it was not possible to perform this evaluation.

The first step was to prepare a V&V plan to describe the process and criteria by which the V&V was to be performed. The second step was to evaluate the code development control procedures used to develop, implement, and maintain SAPHIRE. The third and fourth steps represent the major effort of the project—the design and performance of the software tests.

SAPHIRE features were evaluated and either designated as vital or nonvital features. Vital features are those features essential for completing a PRA analysis such as fault tree analysis, event tree analysis, or uncertainty analysis. For vital features, test cases were developed that specified input, execution conditions, and test criteria. Nonvital features are those features whose malfunction or lack of function will not preclude obtaining results from the SAPHIRE codes. Nonvital features include fault tree construction, event tree construction, and database manipulation. Lack of a nonvital feature would increase the user time required because the work done by that feature would have to be done by other means. Nonvital features were tested for functionality and recorded in a checklist fashion. For both vital and nonvital features, anomaly reports were generated for discrepancies between expected and actual code operation.

This report documents the V&V of SAPHIRE Version 5.0. The nonvital features were tested on the prerelease version of SAPHIRE 5.0 dated October 21, 1993. The vital features were tested on the production version of SAPHIRE 5.0 released to the Energy Science and Technology Software Center on July 20, 1994. Even though the nonvital features were tested on an earlier prerelease version of SAPHIRE, the nonvital features were not retested on the later production version because nonvital features do not affect the accuracy of the numerical results.

To determine the date of a particular version of SAPHIRE 5.0 on a computer, the user can enter the TOOLS subdirectory, type dir *.exe, and press <Enter>. This will produce a list of files with the creation date given.

Previous SAPHIRE V&V efforts include the V&V of SAPHIRE 4.0. The V&V of SAPHIRE 5.0 is based on this previous effort with the necessary modifications to incorporate lessons learned from the previous effort and to reflect the new SAPHIRE 5.0 features. For example, the test cases from the V&V of SAPHIRE 4.0 were revised to account for changes made to existing features and new test cases were added to account for the new SAPHIRE 5.0 features.

3. SAPHIRE V&V PLAN

The SAPHIRE V&V plan was prepared for the purpose of describing the process and criteria by which the V&V would be performed. The SAPHIRE V&V plan was prepared in accordance with IEEE "Standard for Software Verification and Validation Plans."[2] This standard provides uniform and minimum requirements for the format and content of software V&V plans. Even though this standard applies to all phases of the software life cycle from the concept phase to the operation and maintenance phase, the SAPHIRE V&V plan was tailored to apply to SAPHIRE being in the operation and maintenance phase of the software life cycle.

The SAPHIRE 5.0 V&V plan, provided in Appendix A, discusses such topics as the purpose, other binding compliance documents, and an overview of the V&V, including top-level discussions of the V&V tasks that will and will not be performed, as well as the software features to be tested. The V&V plan also addresses reporting and administrative procedures.

4. EVALUATION OF CODE DEVELOPMENT CONTROL PROCEDURES

One of the tasks for this V&V effort was to review the code development control procedures. The first major step of this task was to determine what the minimum code development control procedures should be for any code no matter what development methodology is used (waterfall, cyclic, rapid prototyping, etc.). To do this, a minimum set of V&V activities was identified from IEEE "Standard for Software Verification and Validation Plans" and the NRC *Software Quality Assurance Program and Guidelines*.[2,3] The inputs to performing these minimum V&V activities can then be viewed as representing the minimum code development control procedures. For example, some of the potential minimum V&V activities could be evaluation of the software requirements document, performing a cross-reference evaluation between the requirements document and the design document, and reviewing the configuration management procedures. The inputs, and thus the minimum code development control procedures, would be such items as developing and documenting the software specifications and design, and establishing and implementing configuration management procedures. The second major step of this task was then to evaluate the actual code development control procedures that were used for SAPHIRE 5.0 against the minimum set and make any necessary recommendations.

It should be noted that deriving a minimum set of code development control procedures based on V&V activities is only one approach that can be used. It is possible that a different set of minimum code development control procedures could be derived based on another approach such as project management activities. Even though different approaches were not tried, it is judged that the set identified would always show up as minimum code development control procedures for any software development project. It is not recommended that this "minimum set" take the place of existing standards and guidelines. For any software project, the standards and guidelines should be reviewed and evaluated so that procedures can be established that are commensurate with the importance of the code.

Section 4.1 gives a very brief overview of the software development life cycle and variations that are commonly used. In Section 4.2 the minimum V&V activities for each life-cycle phase are discussed and from this the minimum code development control procedures are identified. In Section 4.3, the actual code development control procedures used for SAPHIRE 5.0 are evaluated. Conclusions and recommendations are provided in Section 4.4.

4.1 Software Development Life Cycle

The development of a software product can be viewed as progressing through various phases that describe the life cycle. The five phases that are generally used to describe the life cycle are:

1. Requirements analysis
2. System design
3. Implementation

4. Testing

5. Operation and maintenance.

This life cycle is typically referred to as the waterfall approach because one step logically leads to the next. The product developed during one phase becomes the basis upon which the product of the next phase is developed. There are several adaptations of the waterfall life-cycle approach, but the activities performed under each approach are very similar. For example, the life cycle described in NUREG/BR-0167[3] consists of seven phases where testing has been split into (a) qualification testing and (b) installation and acceptance, and maintenance has been split into (a) operations and sustaining engineering and (b) retirement and archiving. However, the overall activities that are performed in both life-cycle approaches are the same.

4.1.1 Waterfall Life-Cycle Phases

The following briefly describes each life-cycle phase. Following this is a brief examination of some adaptations to the waterfall life cycle.

Requirements Analysis. Requirements are gathered and analyzed during this first phase. The development team interviews the sponsor (as used in NUREG/BR-0167[3]) and users, individually and collectively, to determine what the proposed code is to do. The purpose of requirements gathering and analysis is to identify the desired functionality of the system. The development team documents these requirements in some formal format, usually following a predefined standard. If the standard chosen is IEEE, the requirements document will be identified as the System Requirements Specification (SRS).

Formal and informal requirements reviews will be held to allow the sponsor and user community to evaluate the development team's understanding and interpretation of their needs, wants, and desires. Reviews are held to determine if the requirements are consistent, complete, correct, and most importantly, what the sponsor really wants. A final requirements review is held that allows the sponsor and user community to formally accept the requirements documentation. Formal acceptance of the requirements identifies the baseline requirements for the remainder of the development efforts. The requirements document should then be placed under configuration control. Further modifications to the requirements document should then follow the software configuration management procedures. The requirements document serves to document this phase of the life cycle and is used in preparing a design document during the next phase.

System Design. This phase involves determining how the system is to be designed in order to satisfy the requirements identified during the previous phase. The development team will put together the algorithms needed to accomplish the functionality requested. The design may be accomplished through the use of a variety of tools including computer assisted software engineering (CASE) tools or prototypes. The design phase will also be documented by the development team. If using the IEEE methodology, the design will be documented in a Software Design Description (SDD). During this phase, design reviews are held where the development team presents the design to the sponsor and user community for input and approval. The reviews may be formal or informal with the final review meeting being a formal meeting held for the purpose of the sponsor/user community to approve the finalized design document.

During design reviews, requirements may be identified that have not been satisfied by the design. There may be several reasons for this such as the requirement being overlooked, conflicting with another requirement, or hardware limitations. If the requirement cannot be satisfied, it will be necessary for the development team to modify the requirements document per configuration management procedures.

Implementation. This phase involves generating the code to satisfy the requirements as described in the design. The requirements analysis phase determined what was to be done. The design phase identified how it was to be done. In this phase, the code is generated to implement the requirements through the design. The development team will test the code produced through shop testing (also referred to as informal unit and integration testing in NUREG/BR-0167[3]). This testing will reveal defects that can be corrected immediately. If the defects impact the requirements or design, it will be necessary to revise the requirements and/or design documents per configuration management procedures. In addition, peer reviews and code walkthroughs will occur. In both activities, the developer's peers assist in evaluating the logic, the correctness of the code, and if the code satisfies the requirements and design to be implemented.

Testing. Testing includes those activities that determine if the generated code satisfies and complies with the requirements and the design. The code is examined through actual execution of the code as well as by using static analysis of the code. Both the development team and the V&V team perform testing.

Testing performed may include but not be limited to:

- *Unit Testing*—Testing performed as a distinct, executable unit is completed to determine if the unit satisfies requirements and correctly implements the approved design.

- *Integration Testing*—Testing of units as they are integrated into larger modules. This testing is directed at the interfaces between previously tested nonintegrated units or modules.

- *System Testing*—When all modules are completed and integrated into a single system, the entire system is tested.

- *Acceptance Testing*—When the system is deemed ready for sponsor/user community use, it is tested by the sponsor/user community to determine compliance to critical functional requirements. Acceptance testing is formal with documented test criteria, test plan, and test results. Successful completion of this testing activity signifies acceptance of the system and willingness to place it into a production environment for use.

Static analysis of the code involves the use of software to collect software metrics or indicators without actually executing the code. Static analysis is also used to assist the test developer in developing test cases and test procedures to accurately and effectively test the code.

Operation and Maintenance. This phase of the life cycle includes the day-to-day activities involving continued use of the system after it has been accepted by the sponsor and entered into a production environment. It usually involves fine tuning the system as the user community becomes more familiar with the system through daily use.

Except for minor changes, system maintenance usually does not include modifications, enhancements, or additions to the code (these are treated the same as development activities). Maintenance activities may include generating reports, monitoring system use and performance, etc.

4.1.2 Adaptations of the Waterfall Life Cycle

Some have interpreted the waterfall life-cycle approach to imply that one phase must be completed before the next can begin. The phases are viewed as a stream of water passing over a waterfall. One rock is covered before the next can be reached. However, the waterfall approach does not dictate nor imply that one phase must be completely finished before the next can begin. As mentioned in the previous section, as the design evolves or as the software is implemented, situations such as an inaccurate requirement, conflicting requirements, or hardware limitations may be identified. Under these situations it will be necessary to return to one or more previous phases to correct, delete, or enhance the requirements document, the design document, or the actual program code.

No phase is totally dependent upon completion of the previous phase. As a part of the code development process, a design team may begin developing some algorithms while the requirements gathering team is still conducting interviews. To insist that each phase be 100% completed and every detail of the product produced for that phase be unchangeable is unrealistic. Each of the previous phases can be readdressed as the development progresses.

There are variations of the waterfall life cycle, but mostly in descriptions of what occurs within each phase. The phases will usually be repeated through a cyclic process. The following describes a couple of variations on the life-cycle approach.

Whirlpool Life Cycle. As the name implies, the whirlpool life cycle circulates through a series of activities. These activities include requirements analysis, design, implementation, testing, and maintenance. The phases are the same as those identified in the classic waterfall software development methodology. The significant difference between the two lies almost entirely in the approach in which the phases are encountered. The whirlpool model predicts and recommends a development approach where the products of each phase are dynamic. They may be changed during activities of any successive phase. Changes will require a return to the present or previous phase and the activities performed therein. Products produced in earlier phases can be reexamined as the development progresses.

This methodology encourages the return to earlier phase activities if the product of that phase can be improved in such a manner as to improve the products produced during activities of later phases. The iterative process of following the waterfall life-cycle phases, when diagrammed, present the appearance of a spiraling whirlpool through the activities of all phases. As the product of each phase is repeatedly improved, the cycle becomes tighter and tighter until the project is completed and the product delivered.

Rapid Prototyping. Rapid prototyping should not be viewed as a software development life cycle. It is a tool to be used during requirements analysis and design to further clarify and refine requirements and design issues. Rapid prototyping may be successfully implemented into a variety of different life-cycle approaches.

During rapid prototype development, the development team prepares models of the proposed system. These models are evaluated by the sponsor and user community to determine compliance to the requirements and to assist developers in designing the system. The model is a tool through which the developers can demonstrate their understanding of desired system functionality and through which the sponsor and user community can clarify their needs and expectations. Rapid prototyping is the process of quickly building and evaluating the requirements and specifications for the critical components of the desired system. The sponsor, user, and designer work together to define the requirements and the specifications.

Rapid prototyping serves as an aid in analysis and design. The prototype is not production software. It is used to redefine and/or clarify system specifications. The designer uses the validated requirements as a basis for continuing design activities and for designing the production software. Documentation prepared during the development of the software serves to assist in establishing a baseline from which the next prototype is developed.

The development of software through rapid prototyping does not preclude the necessity of developing correct, complete, and consistent requirements nor does it preclude the necessity to document requirements and design in formal documentation. Rapid prototyping also involves each of the five phases of the classic waterfall software development life cycle. Requirements are still collected and documented. Design issues are clarified through prototyping desired system behavior. A prototype demonstration may be held in lieu of a review to determine adequacy of design or to help clarify requirements issues. Units are tested upon completion and the modules are integrated into a whole system and tested. The sponsor and user community perform acceptance testing before accepting the finished product. Finally, the code moves it into the production environment and the project enters the maintenance phase.

4.2 V&V Activities and Code Development Control Procedures

The approach of developing software by engineering quality into the product as it is developed is preferred to an approach that attempts to test in quality, by testing out bugs, after the software has been developed. The set of activities designed to ensure that each phase of the development of a software product conforms to the requirements established in the previous phase is often defined as Software Quality Assurance (SQA). Projects that design and implement a specific set of SQA policies and procedures are more likely to produce software that complies to the stated requirements regardless of the software development approach and the language or tools used to develop the system. Software products that have included specific V&V activities at each phase will have a lower error rate and higher confidence level that the software will perform as desired.

V&V should actually be an integral part of any software product development. V&V activities are designed to be performed during each phase of the life cycle to increase the probability that defects and errors will be detected and corrected early in the life cycle. Early detection implies early correction at a fraction of the cost to correct it later. V&V activities can be performed by the development team, by an independent V&V team, or by both teams. For example, the requirements and design reviews can be viewed as a V&V activity because these reviews allow all parties (sponsor, users, and development team) to check the consistency and correctness of the requirements and design. An evaluation of the requirements and design documents by an independent V&V team is also a V&V activity and acts as another check.

Evaluation of Procedures

V&V activities implemented during the development (including modifications, enhancements, or additions) of a software product involve more than just a set of testing activities to be performed during or after the actual development of the code. To ensure that a completed software package performs the desired tasks in the prescribed fashion and attains the desired results, the entire development process for the code needs to be examined. To do this, intermediate products are examined and evaluated at each stage or phase of the development of the software product. The following sections provide a brief overview of the minimum V&V activities that should be performed at each phase of the life cycle, and from this identify the resulting minimum set of code development control procedures.

Requirements Analysis Phase. During this phase the V&V team would review the requirements document for consistency and completeness. If any discrepancies are identified, anomaly reports would be generated and submitted to the development team. The requirements document is also critical in the preparation of test cases, test data, and test procedures to demonstrate system compliance to requirements. Without a formally developed set of requirements, it is very difficult to demonstrate system compliance to stated requirements. The requirements document would be used by both the development team and the V&V team to start putting together tests cases. The V&V team would also review the configuration management procedures during this phase.

The inputs needed to perform the above V&V activities are a requirements document and configuration management procedures. Therefore, the minimum code development control procedures are developing and documenting the software requirements and establishing configuration management procedures.

System Design Phase. During this phase the V&V team would evaluate the design document for compliance and consistency both within the design document and with the requirements document. The evaluation of the design document should identify any requirement that has not been satisfied and if each section of the design document can be justified by a requirement. An anomaly report would be generated for any discrepancies. If the requirement cannot be satisfied, it will be necessary for the development team to modify the requirements document per configuration management procedures. Because it may be necessary to revise the requirements document, how well the configuration management procedures are implemented may also be reviewed by the V&V team.

As with the requirements document, the design document is critical to the development of test cases, test data, and test procedures to be used in testing the system. Requirements are used in preparing tests to demonstrate compliance to system requirements. Design documentation is used to develop tests to exercise such items as each logic path and each decision statement in order to provide a predetermined level of code or path coverage. Without design documentation, this type of testing is very difficult to perform.

The inputs needed to perform the above V&V activities are a design document and configuration management procedures. The code development control procedure for this phase would be developing and documenting the design. Establishing configuration management procedures was identified in the previous phase.

Implementation Phase. The minimum V&V activity during this phase would be to compare the source code against the requirements and design documents to determine if the code implements

the design and complies with the requirements. Any discrepancies would result in an anomaly report. As in the previous phase, there may be cases where a particular part of the design or a requirement cannot be implemented. This may necessitate the modification of the requirement and/or design documents.

Even though there are minimum V&V activities that occur during this phase, the inputs are still a requirements document and a design document. Therefore, there are no new code development control procedures for this phase.

Testing Phase. Both the development team and the V&V team perform testing. The development team will usually perform unit, integration, and system testing first to identify any problems. The development team may also have a group of users perform beta testing. When the development team has determined that the code exists in a state that is essentially ready for production, it is placed under version control and submitted for acceptance testing. For the purposes of the V&V, the level of testing is determined beforehand. The V&V team may also perform unit, integration, and system testing, as well as acceptance testing. Test cases that were prepared during the previous phases when the requirements, design, and source code were being evaluated are finalized. Developing test cases through the life cycle provides better assurance that system compliance with the requirements can be demonstrated and that there will be adequate test coverage of the system. Formal reports describing the results of the testing efforts are prepared. The contents and format of the report should comply with adopted standards and guidelines. A detailed test log is usually maintained during test efforts that identifies the item being tested, the actions to be performed by the tester, the expected results, and the actual results obtained during testing. Any discrepancies are logged.

As with the previous phases, the completeness of the testing is highly dependent on having established and documented requirements and design information. Even though testing really is not an "input" as this term has been used in the previous phases, establishing and documenting a testing program that covers testing that will be performed by the development team and by the V&V team is also considered to be a minimum code development control procedure.

Operation and Maintenance. If version control has not previously been addressed in the configuration management procedures, it should be before the code is placed into production. A system for dealing with nonconformance reporting and correction should also be established and coordinated with the configuration management procedures so that control of the software and code version can be controlled. A potential V&V activity during this phase would be to evaluate the procedures for version control and for nonconformance reporting and correction. Therefore, the minimum code development control procedures include establishing version control and a system for nonconformance reporting and correction, with both being a part of or coordinated with the configuration management procedures.

4.2.1 Additional Guidelines

NUREG/BR-0167[3] identifies two other guidelines that need to be addressed. These two guidelines do impact the minimum code development control procedures.

Software Level. NUREG/BR-0167 identifies three levels of software:

1. *Level 1 Software*—Technical application software used in a safety decision by the NRC (an example would be RELAP5)

2. *Level 2 Software*—Technical or nontechnical application software not used in a safety decision by the NRC (an example would be an agency financial software system)

3. *Level 3 Software*—Technical or nontechnical application software not used in a safety decision and having local or limited use by the NRC (an example would be a macro for Lotus 1-2-3).

The guidelines presented in NUREG/BR-0167 apply to Level 1 and Level 2 software. It is stated that the degree of applicability of the guidelines will depend on such items as the purpose and use of the software. The level of the software is designated by the sponsor.

The determination of the software level is related to V&V activities and code development control procedures. The extent of the V&V and the necessary code control procedures can be better established if the software level is known. Therefore, an additional minimum code development control procedure for NRC software would be establishing the software level.

Applicability of NUREG/BR-0167. It should be noted that NUREG/BR-0167 was issued in February 1993. It is recognized in NUREG/BR-0167 that all of the detailed guidelines cannot be applied in a cost-effective manner to software developed before this document was issued; however, NUREG/BR-0167 recommends that the following actions be taken on a step-by-step basis to enhance existing software:

1. Establish and maintain a software development library

2. Institute a nonconformance and corrective action system

3. Develop a set of acceptance test cases

4. Institute a clearly defined test program

5. Institute a well-defined configuration management system, especially a software release system

6. Begin code inspections.

Establishing and maintaining a software development library was assumed to correspond to the previously discussed code development control procedures of developing and documenting the software requirements and the design. The institution of a nonconformance and corrective actions system and of a well-defined configuration management system that includes a software release system have also been previously discussed as a minimum code development control procedure. The development of a set of acceptance test cases and the institution of a clearly defined test program were also identified as a minimum code development control procedure. It was assumed that code inspections, which have been identified as a V&V activity during the implementation phase, referred to comparing the source code to the requirements and design documents. The difference is due to

identifying code development control procedures on the basis of a new code development or an existing code that is still undergoing enhancement and modifications rather than on the basis of an existing code that is relatively static. Except for the code inspections step, the minimum code development control procedures that were previously identified are considered to correspond to the actions recommended by NUREG/BR-0167 for existing software.

4.2.2 Summary of Minimum Code Development Control Procedures

The above has identified that the minimum code development control procedures for NRC software are essentially:

- Establishing the software level
- Developing and documenting the software requirements
- Establishing configuration management procedures that include or have been coordinated with code version control and nonconformance reporting and correction
- Developing and documenting the design
- Establishing and documenting a testing program that includes independent acceptance testing.

4.3 Evaluation of SAPHIRE 5.0 Code Development Control Procedures

4.3.1 Software Level

The software level for SAPHIRE has not been established because versions of this code up to and including SAPHIRE 5.0 were already developed or planned before NUREG/BR-0167 was issued. It is recommended for future versions of SAPHIRE that consideration be given to designating the software level and to developing a program that will institute on a step-by-step basis the suggested actions to enhance existing software (as discussed under the "Applicability of NUREG/BR-0167" subsection in Section 4.2.1).

4.3.2 Develop and Document Software Requirements

The new features, enhancements, and/or modifications to be included in the next version or release of SAPHIRE are determined by the NRC. The SAPHIRE development team maintains a list of sponsor/user requests for system modification. The list is submitted to the NRC for prioritization. The NRC selects those items deemed important for continued use of SAPHIRE and to be included in the next version or release.

A project plan (as represented by the NRC Form 189) is prepared that describes the new features and/or modifications to be included in the next release. The project plan serves as a high level requirements document for continued efforts on SAPHIRE. No detailed requirements document is prepared at this stage of the development process.

For SAPHIRE 4.0, the requirements were documented in *Integrated Reliability and Risk Analysis System (IRRAS), Version 4.0, Functional Requirements Document* (Draft).[a] However, for SAPHIRE 5.0, the requirements were not documented. Because no documentation of the requirements was prepared, it was not possible to review the requirements for completeness, correctness, consistency, or quality. The lack of documented requirements also limits the ability of the sponsor/user community to perform thorough acceptance testing or to formally determine if the project has been successfully completed. From a review of the project plan (L1429 SAPHIRE Maintenance and User Support), it was determined that there was no requirement for the SAPHIRE 5.0 requirements to be developed and documented.

For future versions of SAPHIRE it is recommended that detailed software requirements be developed and documented. Even though it was identified in the V&V of SAPHIRE 4.0[1] that some improvements were needed to the IRRAS 4.0 Functional Requirements Document, it did serve as a first step in the right direction of establishing a software development library. The life-cycle methodology described by the development team is best identified as the whirlpool life cycle. Because the whirlpool life-cycle methodology is used for SAPHIRE development purposes, it is viewed that little additional effort and resources would be required for formal documentation of the requirements.

It is also difficult to derive a complete, consistent, and correct set of requirements without sponsor/user involvement in reviews of the requirements. It is recommended that consideration be given to conducting requirements reviews to help clarify and finalize the requirements. Once the new features, enhancements, and/or modifications have been chosen for the next version of SAPHIRE, a small group of users could be gathered for the purpose of providing initial input on how these items should work from a PRA perspective. The development team takes this input and translates it into how the code will work. Prototyping may be used as a tool to put together an initial model of that feature. The small group of users could then be gathered again to evaluate the prototype of the feature. Comments would be gathered on agreed upon changes to the prototype. Requirements would be generated based on the initial input and on the changes that were agreed to from exercising the prototype. The same process would be used for each feature (enhancement, modification, etc.) with the end result being the requirements for the new version of the code. At a later date the prototypes and requirements for each feature would serve as the models for making changes to the baseline source code.

4.3.3 Configuration Management

Configuration Management Procedures. Overall, configuration management complies with the requirements specified in the Conduct of Operations Manual for the Scientific Computing Unit.

Version Control. Released versions are assigned an appropriate release number and placed under version control as specified in the Conduct of Operations Manual. However, interviews with SAPHIRE users indicate that the users are not aware of the difference between beta test versions and versions released for production use. This misunderstanding has given numerous users the perception that the SAPHIRE project has loose or nonexistent version control policies and procedures in place. However, the beta versions are clearly labeled as such. The actual problem

a. By S. Ted Wood and Kenneth D. Russell, EG&G Idaho, Inc., Idaho Falls, Idaho, December 1991.

appears to be due to using beta test versions on projects that should actually be using only production versions. The credibility of SAPHIRE is being significantly affected by this practice.

Nonconformance Reporting and Corrective Action. The SAPHIRE project does maintain a log of defect reports, including resolution to those defect reports.

4.3.4 Develop and Document the Design

No separate design document was prepared for SAPHIRE 5.0. From a review of the project plan (L1429 SAPHIRE Maintenance and User Support), it was determined that there was no requirement for the SAPHIRE 5.0 design be developed and documented. For future versions of SAPHIRE, it is recommended that consideration be given to holding design reviews and to documenting the design. Design reviews allow the development team to present to the sponsor and the user how the requirements are going to be implemented. A prototype model for each feature or group of features is especially useful for design reviews because it gives the sponsor and users a visual product that demonstrates the design. Documentation of the design would also define the software units and modules that are important for the purposes of unit and integration testing.

4.3.5 Establish and Document a Testing Program

For SAPHIRE 5.0 the implementation and testing phases are very closely intertwined due to using the whirlpool life-cycle approach in conjunction with beta testing. The requirements phase is also revisited during the implementation phase, with the number of times each phase is repeated dependent entirely on the beta test results.

During the implementation phase, further clarification of the requirements is obtained through correspondence and conversations with the sponsor/user requesting a certain feature. As the requirements are clarified, the system is developed and coded. When a module appears to have satisfied the requirements collected by the development team, a beta version is released to a set of users for testing. For SAPHIRE 5.0, the majority of the beta testing has been essentially performed under a separate project, Plant Database Development for SAPHIRE (W6241). The version is assigned a version number and is placed under version control procedures as outlined in the Conduct of Operations Manual.

The beta testers report any discrepancies and findings to the development team. The development team then implements any indicated modifications. When it appears that the modified version better "fits" existing system requirements, another beta version is released to the beta testers. This process is continued until a version is produced that appears ready to comply with the system requirements as described in the project plan. This version is then placed under version control, with an appropriate version number assigned and released for production purposes to the sponsor and user community.

The process of identifying requirements, clarifying requirements, developing code to implement the requirements, and performing beta testing is repeated until the sponsor formally accepts the released version as an accepted product.

Evaluation of Procedures

The V&V of SAPHIRE is a separate project that includes acceptance testing as one of the tasks. Once it appears to the V&V team that the beta testing is no longer identifying major problems and that the code appears fully functional, the V&V team takes the version of the code at that time and initiates acceptance testing. This version of the code is used throughout the acceptance testing. The independent V&V project budget and schedule do not accommodate an iterative cycle of performing the acceptance testing and producing any anomaly reports, the development team updating the code, and the V&V team repeating the acceptance tests.

Tests have been developed for the purpose of determining if the results produced by the code are correct, and if not, to identify any areas requiring correction. Specifically, under the V&V of SAPHIRE 4.0 test cases were developed by experienced users to test vital and nonvital features of the code. For the V&V of SAPHIRE 5.0, experienced users revised and modified these tests and developed additional tests so that the test cases would be applicable to SAPHIRE 5.0. More stress tests were added to the test cases for SAPHIRE 5.0 so that known limits could be tested and unknown limits could be identified. Vital features are defined as those features that (a) affect the results of a PRA (core damage frequency, cut sets, etc.) and (b) are essential for completing a PRA (fault tree analysis, event tree analysis, uncertainty analysis, etc.). Nonvital features are defined as those features whose malfunction will not preclude getting the final results from the code but may result in additional time to perform the analysis because the work that would have been done by that feature will have to be performed in another manner.

Where possible, test results will be compared to theoretically-based hand calculations and/or to results from the SAPHIRE 4.0 test cases. A successful comparison of the 5.0 results to the 4.0 results will provide added confidence that the enhancement/modifications made to produce SAPHIRE 5.0 have not degraded the performance of those portions of the code that were not changed. Where hand calculations are not practical, existing codes such as CAFTA and PC-SETS software will be used to provide computer-assisted results from comparison. Although neither CAFTA and PC-SETS have been verified and validated, it is deemed appropriate that correlation of results with SAPHIRE signifies correct operation of the code because the three codes use different coding techniques and methodologies.

A major new feature for SAPHIRE 5.0 is the ability to perform seismic analysis. To test this feature, the results of a task from another project will be used. Under the project W6241 Plant Database Development for SAPHIRE, there is a task to load the Surry seismic analysis into SAPHIRE 5.0 and compare the results against the NUREG-1150 results as documented in *Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events*.[4] The successful comparison of the results constitutes the acceptance criteria for the seismic analysis function.

Even though the above independent V&V testing program is a good effort, it should be noted that it has been necessary to prioritize the tests due to schedule and budget constraints. For the SAPHIRE 4.0 V&V, all high priority tests were performed. It is expected for the SAPHIRE 5.0 V&V that it will only be possible to perform all high priority tests, including testing the seismic analysis feature. In addition, without documented requirements and design, it is not possible to determine if the test cases are adequate and thoroughly test the SAPHIRE code. As was identified in the V&V of SAPHIRE 4.0, only through an analysis of requirements, design, and source code can adequate testing be designed and performed. Finally, it should also be noted that for both the V&V of SAPHIRE 4.0 and 5.0, acceptance testing has occurred after the code has been released for

production purposes. As was the case with the SAPHIRE 4.0 V&V, documentation of the V&V results does provide the users with information on what features do not properly work for a specific version of the code.

4.4 Conclusions and Recommendations

Due to the complex nature of the SAPHIRE code and the importance of the results obtained from executing SAPHIRE, there are improvements that could be made in both the development and V&V areas that would help ensure that the results the code produces are correct.

It is recommended that consideration be given to establishing the software level for SAPHIRE so that the development and V&V activities for the code are commensurate with the software level.

For major modifications and enhancements, it is recommended that consideration be given to holding a requirements gathering meeting and at later dates holding requirements and design reviews. The requirements gathering meeting can provide the development team with an excellent opportunity to get the users to describe in detail all of the steps that a PRA analyst goes through when performing a certain PRA process. The users should understand that they are not there to perform the coding—they are there to describe the PRA process in sufficient detail so that the development team can translate the process into coding that will perform the process in a more automated manner. The descriptions of the process form the basis for the requirements. The later requirements and design reviews provide the development team with the opportunity to check their interpretation of the process.

It is recommended that the requirements and design be formally documented. Without formally documented requirements and design, it will not be possible for a testing program to be comprehensive.

It is recommended that consideration be given to coordinating the development and V&V projects so that there is a more comprehensive testing program that occurs over the entire life cycle. The testing program should address the type of testing (unit, integration, system, and acceptance) that will be performed and the test documentation requirements.

Generally, acceptance testing is performed before the sponsor accepts the code for production purposes. It is recommended that consideration be given to coordinating the development and V&V projects so that acceptance testing is performed before the code is released to the Energy Science and Technology Software Center; however, it should be recognized that this can result in the need for iterative acceptance testing. To limit the number of iterations of acceptance testing (if it is decided to perform acceptance testing before release of the code for production use), it is recommended that once the development team has completed internal testing and beta testing and is satisfied that the code is ready for production use, that it be formally transferred to the V&V team for acceptance testing.

Finally, it is recommended that beta test versions not be used on projects that should be using production software. Not only is the credibility of the SAPHIRE code being affected by this practice, but the budgets and schedules of these other projects are also being impacted.

5. VITAL FEATURES TEST DEVELOPMENT

SAPHIRE features were evaluated and designated either as vital or nonvital features. Vital features are those features that (a) affect the results of a PRA (core damage frequency, top contributors, etc.) and (b) are essential for completing a PRA analysis (such as fault tree analysis, event tree analysis, uncertainty analysis, etc.). The overall purpose of the vital features testing was to determine if the results produced by SAPHIRE 5.0 are correct, and if not, to document those areas requiring correction.

How the vital features and test procedures were established is discussed in the following section. To evaluate the completeness of the test procedures, user interviews were conducted to determine if the initial test procedures adequately covered problems that users had encountered in their use of prerelease version of SAPHIRE 5.0. The results from the user interviews is also presented.

5.1 Establishment of Vital Features

Under the V&V of SAPHIRE 4.0, vital features and the tests for these vital features were developed by experienced SAPHIRE users. For the V&V of SAPHIRE 5.0, experienced users have revised the tests as needed to reflect modifications made to existing features and have developed additional tests to reflect the new features that have been added to SAPHIRE 5.0. Additional boundary tests have also been included so that known limits could be tested and unknown limits could be identified. The testing did not include such items as entry of negative or bogus numbers, typing in words where numerical data are expected, or of the ability of SAPHIRE to recover for power failures or similar system failures.

The identification of vital features was developed by outlining the major functions that are performed in a PRA. These include fault tree analysis, event tree analysis, uncertainty analysis, change set features, cut set editor, and importance measures. These functions are shown in the first column of Table 1. Vital features were then determined for each of these functions. The vital features are shown in the second column of Table 1 and include such items as cut set generation, quantification, event tree sequence generation, and applying data change sets. Subfeatures for each vital feature were developed to identify what aspect of the vital feature was to be tested. For each subfeature, a test criterion was developed. Columns three and four in Table 1 present this information. Stress tests are designated in column five.

With this process, 73 tests were identified as is shown in column six of Table 1. It was recognized that performing all 73 tests would not be possible within the time available for this project. Under the V&V of SAPHIRE 4.0, all of the subfeatures had been prioritized as high, medium, or low by experienced users and all high priority subfeatures were tested. For the V&V of SAPHIRE 5.0, it was decided that all high priority subfeatures would again be tested first. A major new feature for SAPHIRE 5.0 is the ability to perform seismic analysis, and testing of this feature was also designated as high priority. The tests that were actually performed for the V&V of SAPHIRE 5.0 are designated in column seven.

Table 1. SAPHIRE 5.0 vital features.

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
Fault Tree Analysis	Cut set generation process	Cut set generation for a small fault tree(s) (can be quantified by hand)	Determine if the correct cut sets are generated. The fault tree(s) will consist of the following gate types: AND, OR, N/M, transfers, and complemented. The fault tree(s) will also consist of an "X" Process Flag with both independent and dependent basic events.	N	1	Y
		Cut set generation for a large fault tree	Determine if the correct cut sets are generated. The fault tree will consist of the following gate types: AND, OR , N/M, transfers, and complemented (if CAFTA can handle). The fault tree will also consist of an "X" Process Flag with both independent and dependent basic events.	N	2	Y
		Cut set generation for a large fault tree	Determine if the correct cut sets are generated. The maximum number of gates for a fault tree is 9,999. The fault tree for this test will have a minimum of 1,500 gates. The fault tree will consist of the following gate types: AND, OR, N/M, transfers, and complemented (if CAFTA can handle). The fault tree will also consist of an "X" Process Flag with both independent and dependent basic events.	Y	3	Y
Fault Tree Analysis	Cut set quantification process	Cut set probability cutoff	Determine if the only cut sets kept are those with a probability that is above the cut set probability cutoff.	N	4	Y
		Event probability cutoff	Determine if the cut sets kept are those where each event in the cut set is above the probability cutoff.	N	5	N
		Starting gate	Determine if the correct cut sets are generated from the logic below the designated starting gate.	N	6	Y
		Maximum number of cut sets	Determine if the maximum number of cut sets saved exceeds the memory capacity of the computer used for this test.	Y	7	Y
		Size truncation	Determine if cut sets are correctly truncated based on the number of events in the cut set.	N	8	N
		Zone truncation	Determine if cut sets are correctly truncated based on the number of zones in the cut set.	N	9	N
Fault Tree Analysis	Cut set quantification process	Minimal cut set upper bound for alternate cut sets	Generate the minimal cut set upper bound for alternate cut sets of the selected systems. The upper bound is the bounding number for the sum of all the cut sets. Determine if the upper bound is correct.	N	10	Y
Fault Tree Analysis	Cut set quantification process	Min-cut set upper bound	Generate the base case cut set upper bound using the min-cut upper bound equation. Determine if the upper bound is correct. The equation for the min-cut upper bound is	N	11	Y

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			$MC = 1 - \prod_{1}^{m} i (1 - C_i)$ where MC = minimal cut set upper bound C = probability of the i^{th} cut set m = number of minimal cut sets in the sequence.			
		Min-max quantification	Determine if the cut set quantification matches the exact calculation.	N	12	N
Fault Tree Analysis	Editing and modifying of fault tree cut sets	Cut set editor —list all systems and allow user to select a system to edit its base cases or alternate cut sets	Determine if the following features work correctly: a. Add a cut set—add a brand new cut set b. View an existing cut set c. Modify cut set—change the events that exist in a cut set d. Delete cut set—delete the entire cut set e. Restore cut set—restore the last deleted cut set f. Add event to cut set—add event to existing cut set g. Modify event in cut set—modify highlighted event name h. Delete event from cut set—Remove the highlighted event from the cut set i. Restore event—restore the last deleted event j. Find—finds the cut set(s) that contains a given list of events and performs the following functions: Insert: Add the specified event to the cut set Replace: Replace a list of events with the specified event Delete: Delete the found cut set Copy/Add: Create a copy of the found cut set and in that copy, replace the list of events with the specified event.	N	13	N
Fault Tree Analysis	Analyze systems	Prune cut sets	Determine if the system cut sets are correctly pruned given a specified criteria.	N	14	N
Fault Tree Analysis	Recover cut sets	Family system	Determine if the rules make the desired changes to the cut sets.	N	15	N
Fault Tree Analysis	Editing and modifying basic events, gates, fault tree graphics by using Logic Editor	Logic editor	Test the Logic Editor by: Entering a one gate logic file. Check the gate name against the basic event name. Determine if an error message is displayed if a gate name is changed to be the same as the basic event name.	N	16	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			Modify the gate name in one fault tree. Determine if gate name was or was not modified in another tree that uses the same gate name. After making changes in the Logic Editor and performing an Alpha to Graphics Conversion, determine if fault tree graphics logic is the same as the Logic Editor logic.			
Event Tree Analysis	Sequence Process Generation	Generate sequences	Determine if all possible sequences for an event tree are correctly generated. Determine if the logic for each sequence is correctly generated except for those end states marked as "OK," "SUCCESS," "IGNORE," or with the character "@". Determine if a mutually exclusive top can be added and correctly handled.	N	17	Y
Event Tree Analysis	Sequence process generation	Sequence editor—allow user to change the end state header and fields for a given event tree	Determine if the following features work correctly: Frequency—(1) look up sequence minimum cut set upper bound for highlighted sequences (0.0 is default). (2) add frequency to the event tree drawing, (3) change header for the third column to be frequency. @ status—change the end state name to be generated (without @ in the first character) or not generated (with @ in the first character) for each highlighted sequence. Line edit—change any highlighted line. Follow transfer—if transfer event tree exists, edit its logic. Global replace—given the column number and a search string, replace any occurrences of the search string in the given column number with the replace string. Header edit—change the header for the drawing. Change transfer—toggle logic between being a transfer path or a termination path.	N	18	Y
Event Tree Analysis	Sequence process generation	Link editor—allows user to enter links (exceptions) that are to be applied during sequence logic generation	Determine if multiple branches are handled correctly by evaluating the sequence cut sets. This evaluation is to be performed for the cases of "adding," "modifying," "deleting," "restoring," and "copying" a rule. This test should specifically determine that when a sequence is added or deleted, the sequence names and frequency values are correct.	N	19	Y
		Link editor—truncation by probability	Determine if the only sequences kept are those with a probability that is above the sequence probability cutoff.	N	20	N
		Link editor—generate cut sets	Determine if the sequence cut sets are replaced with a single cut set that correctly represents the sequence logic.	N	21	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
Event Tree Analysis	Sequence cut set generation	Cut set probability cutoff	Determine if the only cut sets kept are those whose probability are above the assigned cutoff.	N	22	Y
			Determine if the cut set probability of failure is equal to the product of the failure probability of each event.			
		Event probability cutoff	Determine if all cut sets have been kept whose failure probability is greater than the assigned event probability cutoff.	N	23	N
			Determine if the failure probability for each event in the cut set is greater than or equal to the assigned event probability cutoff.			
		Size truncation	Determine if cut sets are correctly truncated based on the number of events in the cut set.	N	24	N
		Zone truncation	Determine if cut sets are correctly truncated based on the number of zones in a cutset.	N	25	N
		Solve sequence with fault trees—combine the fault tree logic for all referenced successful and failed systems, generate cut sets, and solve using fault tree cut set generation algorithm	Determine if the correct sequence cut sets are generated for the following cases. Check several sequences to determine if system success and failure are being correctly handled.	N	26	N
			(1) Small event tree (can be quantified by hand). The fault trees used by the event tree will consist of the following gate types: AND, OR, N/M, transfers, and complemented. The fault trees will also use Process Flags with both independent and dependent events.			
			(2) Large event tree. The fault trees used by the event tree will consist of the following gate types: AND, OR, N/M, transfers, and complemented. The fault trees will also use Process Flags with both independent and dependent events.			
Event Tree Analysis	Sequence cut set generation	Solve sequence with system cut sets (i.e., use existing cut sets rather than system logic)	Combine the cut sets for all referenced systems to generate the sequence cut sets. Determine if the correct sequence cut sets are generated for the following cases. Check several sequences to determine if system success and failure are being correctly handled.	N	27	N
			(1) Small event tree (can be quantified by hand). The fault trees used to generate the system cut sets will consist of the following gate types: AND, OR, N/M, transfers, and complemented. The fault trees will also use Process Flags with both independent and dependent events.			
			(2) Large event tree. The fault trees used to generate the system cut sets will consist of the following gate types: AND, OR, N/M, transfers, and complemented. The fault trees will also use Process Flags with both independent and dependent events.			
		Large Event Tree (LET)	Determine if the correct sequence cut sets are generated when the split fraction technique is used. Check several sequences to determine if system success and failure are being correctly handled.	N	28	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
Event Tree Analysis	Sequence cut set generation	Maximum number of cut sets saved for one sequence	Determine if the maximum number of cut sets saved exceeds the memory capacity of the computer used for this test.	Y	29	Y
		Default flag set	Each sequence can have a flag set that applies only to it during the cut set generation. Determine if the flag set will correctly overwrite any change sets that are marked. Determine if the keyword "NONE" (or a blank line) will cause cut sets to be generated without any sequence change set.	N	30	Y
Event Tree Analysis	Sequence cut set generation	Update cut sets	Reevaluate existing cut sets based on the current data. Use base case cut sets or alternate case cut sets as the basis for the evaluation. Determine if cut sets have been handled correctly for an assigned cut set probability cutoff. Determine if the probability of failure is equal to the product of the probability of each event in the cut set. Determine if the Size Truncation feature has correctly thrown away any cut sets whose size exceeds a specified size. Determine if the Zone Truncation feature has correctly thrown away any cut sets whose number of zone events exceeds the specified zone size.	N	31	Y
Event Tree Analysis	Sequence cut set generation	Quantify cut sets	Generate minimal cut set upper bound for the alternate cut sets of the selected sequence or sequences. Determine if the upper bound is correct. The equation for min-cut upper bound is $$MC = 1 - \prod_{1}^{m} (1 - C_i)$$ where MC = minimal cut set upper bound C = probability of the i^{th} cut set m = number of minimal cut sets in the sequence.	N	32	Y
		Split fraction	Determine if the split fraction feature works correctly. Calculate the split fraction for the highlighted sequences. The split fraction is an approximation of the minimum cut set upper bound for a sequence. The equation is	N	33	Y

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			$SF=(\prod_{1}^{n}(1-sMC_i))\times(\prod_{1}^{m}fMC_j)$			
			where SF = Split fraction sMC = minimal cut set upper bound of the i^{th} successful system n = number of successful systems fMC = minimal cut set upper bound of the i^{th} failed system m = number of failed systems.			
		Base case update	Determine if the base case update feature correctly works. To determine this, copy the alternative cut set information to the base case. Evaluate the cut sets, quantified values, and uncertainty values to determine if the update was correctly performed.	N	34	Y
Event Tree Analysis	Sequence cut set generation	Quantify cut sets	Generate minimal cut set upper bound for the alternate cut sets of the selected sequence or sequences. Determine if the upper bound is correct. The equation for min-cut upper bound is	N	32	Y
		Split fraction	Determine if the split fraction feature works correctly. Calculate the split fraction for the highlighted sequences. The split fraction is an approximation of the minimum cut set upper bound for a sequence. The equation is	N	33	Y
			$SF=(\prod_{1}^{n}(1-sMC_i))\times(\prod_{1}^{m}fMC_j)$			
			where SF = Split fraction sMC = minimal cut set upper bound of the i^{th} successful system n = number of successful systems fMC = minimal cut set upper bound of the i^{th} failed system m = number of failed systems.			
		Base case update	Determine if the base case update feature correctly works. To determine this, copy the alternative cut set information to the base case. Evaluate the cut sets, quantified values, and uncertainty values to determine if the update was correctly performed.	N	34	Y
Event Tree Analysis	Analyze sequences	Prune cut sets	Determine if the prune logic makes the desired changes to the cut sets.	N	35	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
		End state uncertainty	Determine if cut sets grouped by end states have uncertainty quantified correctly.	N	36	N
		Min-max quantification	Determine if the sequence quantification matches the exact calculation.	N	37	N
Event Tree Analysis	Recover cut sets	Sequence Family Event Tree	Determine if the rules make the desired changes to the cut sets.	N	38	N
Event Tree Analysis	Analyze sequences	Partition cut sets	Determine if the cut sets are correctly partitioned per specified criteria.	N	39	N
Event Tree Analysis	Quantifying process	Alternate cut sets	Determine that the minimal cut set upper bound for the alternate cut sets of the selected sequence or sequences is correctly generated. The equation for min-cut upper bound is $MC = 1 - \prod_{i}^{m} (1 - C_i)$ where MC = minimal cut set upper bound C = probability of the i^{th} cut set m = number of minimal cut sets in the sequence.	N	40	Y
Event Tree Analysis	Capability of performing the analysis on a single event tree	Perform analysis on event tree using fault trees	Determine if the maximum number of events and gates exceeds the memory capacity of the computer used for this test.	Y	41	Y
		Perform analysis on event tree using cut sets	Determine if the maximum number of events and gates exceeds the memory capacity of the computer used for this test.	Y	42	Y
Event Tree Analysis	Editing and modifying of event tree sequence cut sets	Cut set editor—list all sequences and allow user to select a sequence to edit its base case or alternate cut sets.	Determine if the following features work correctly: a. View existing cut sets b. Add cut set—add a brand new cut set c. Modify cut set—change the events that exist in a cut set d. Delete cut set—delete the entire cut set e. Restore cut set—restore the last deleted cut set f. Add event to cut set—add event to existing cut set g. Modify event in cut set—modify highlighted event h. Delete event from cut set—remove the highlighted event from the cut set i. Restore event—restore the last deleted event	N	43	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			j. Find—find cut set(s) that contains a given list of events and perform the following functions: Insert: Add the specified event to the found cut set Replace: Replace the list of events with the specified event Delete: Delete the found cut set Copy/Add: Create a copy of the found cut set and in that copy, replace the list of events with the specified event Exit without saving changes Exit while saving changes.			
End State Analysis	Grouping cut sets according to end state classification	End state partion rules	Determine if the end state partion rules correctly group cut sets according to specified search criteria.	N	44	N
		End state cut set gathering and quantification	Determine if partioned end state cut sets are correctly gathered and quantified.			
Uncertainty Analysis	Single event sampling	Monte Carlo Sampling	Take single event with known distribution and sample the event to determine if the distribution is reproduced for the following: Lognormal Distribution Normal Distribution Beta Distribution Chi-Squared Distribution Exponential Distribution Uniform Distribution Histogram Gamma Maximum Entropy.	N	45	Y
		Latin Hypercube Sampling	Take single event with known distribution and sample the event to determine if the distribution is reproduced for the following: Lognormal Distribution Normal Distribution Beta Distribution Chi-Squared Distribution Exponential Distribution Uniform Distribution Histogram Gamma Maximum Entropy.	N	46	Y
Uncertainty Analysis	Uncertainty analysis of sequences	Monte Carlo Sampling—multiple event sampling	Perform uncertainty on sequences. Check the following levels for the sequences: Single—perform uncertainty on each highlighted sequence individually.	N	47	Y

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			Group—perform uncertainty analysis on a group of highlighted sequences. End State—perform uncertainty analysis on the group of sequences that belong to the same highlighted end state. Family—perform uncertainty analysis on all sequences as a group in the family. System—perform uncertainty analysis on fault tree level. Correlation—perform uncertainty analysis for multiple events where some of the events are correlated.			
Uncertainty Analysis	Uncertainty analysis of sequences.	Latin Hypercube Sampling	Perform uncertainty on sequences. Check the following levels for the sequences: Single—Perform uncertainty on each highlighted sequence individually. Group—Perform uncertainty analysis on a group of highlighted sequences. End State—Perform uncertainty analysis on the group of sequences that belong to the same highlighted end state. Family—Perform uncertainty analysis on all sequences as a group in the family. System—Perform uncertainty analysis on fault tree level. Correlation—Perform uncertainty analysis for multiple events where some of the events are correlated.	N	48	Y
Change Set Features	Creating change sets	Individual Probability Method	Determine if this feature is correctly working by assigning a new probability to one-of-a-kind events and evaluating that each desired change did take place.	N	49	Y
		Tagging Method	Determine if this feature is correctly working by assigning a change definition to a group of events and evaluating that the desired change did take place.	N	50	Y
		Class Method	Determine if this feature is correctly working by establishing a search criteria for a class of events, inputting the desired change definition, and evaluating that the desired change did take place.	N	51	Y
Change Set Features	Applying changes to models	Change set menu	Determine if the code correctly applies changes to models. Enter the change set menu: In IRRAS it is under the "Generate Event Data" menu and in SARA it is under the "Modify Event Data" within the Systems Analysis menu. Mark the change set to be modified and generate the changes. Evaluate if the changes were correctly applied.	N	52	Y
		Apply changes to system data	Determine if the code correctly applies changes to system data. Enter the "Analyze Sequence" menu in IRRAS or the "Analyze Systems" menu or the "Analyze Event Tree" menu in SARA. Select the quantification option and select the system or sequence. Evaluate if the changes were correctly applied.	N	53	Y

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
Change Set Features	Flags to be applied in change set events	Application of the various flags	Determine if the code correctly applies the following flags:			
			Sensitivity Analysis "S" flag—special flag put in for a sensitivity analysis.	N	54	Y
			Do not expand transfers "X" flag—"X" tells SAPHIRE that the basic event is to be used for failure references, but success references are to be treated the same as if the flag was blank.	N	55	Y
			Never expand transfers "Y" flag—"Y" indicates that a transfer is to be replaced with its basic event for failed references and the complement of the event is to be used for success references.	N	56	Y
			Always expand transfers "I" flag—"I" causes SAPHIRE to treat the transfer as independent. Logic below this transfer is expanded for failure references and for success references the complement of the logic is used.	N	57	Y
			Blank flag field—when the process field is blank, the transfer associated with this event is expanded; however, impossible cut sets generated are removed from the cut sets using cut set matching.	N	58	Y
			Key event "K" flag—fails all locks in the same group.	N	59	N
			Lock event "L" flag—failed only by its key event.	N	60	N
			Zoned flagged event "Z" flag—used to designate that the event has location information attached which can be used as a criteria for truncation during cut set generation.	N	61	N
			"W" flag—failure uses system logic. Success uses complemented of developed event.	N		N
Sequence Cut Set Editor		Editing features	Determine if the following features work correctly: a. Add a brand new cut set—view an existing cut set b. Modify cut set—change the events that exist in a cut set c. Delete cut set—delete the entire cut set d. Restore cut set—restore the last deleted cut set e. Add event to cut set—add event to existing cut set f. Modify event in cut set—modify highlighted event name g. Delete event from cut set—remove the highlighted event form the cut set h. Restore event—restore the last deleted event.	N	62	N
Cut Set Editor	List all sequences and allow user to select a sequence to edit its base case or alternate cut sets	Speed search	Determine if the speed search feature works correctly. Enter a name and determine if the first occurrence of that name is highlighted. Verify if the "NEXT" and "PREVIOUS" options work correctly.	N	63	N
		View events with mask	Determine if the Mask feature works correctly. Enter a set of attributes as a mask and determine that the events highlighted do have those attributes. Verify if the "NEXT" and "PREVIOUS" options work correctly.	N	64	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
		Find	Determine if the Find feature work correctly. Find the cut set(s) that contains a given list of events and performs the following functions: Insert: Add the specified event to the cut set Replace: Replace a list of events with the specified event Delete: Delete the found cut set Copy/Add: Create a copy of the found cut set and in that copy, replace the list of events with the specified event.	N	65	N
Importance measures		Fussell-Vessely Risk Reduction Ratio Risk Increase Ratio Birnbaum Importance Risk Reduction Interval Risk Increase Interval	Determine if the following importance measures are correctly calculated. Calculate importance measures with: P(top)—Probability that the top event occurs. P(A)—Probability of event A (the event of interest. P(top/A=1)—Probability that the top event occurs given A always occurs (Probability of A=1). P(top/A=0—Probability that top event occurs given event A never occurs (Probability of A=0) Fussell-Vessely: $$FV=\frac{P(top)-P(top/A=0)}{P(top)}$$ Risk Reduction Ratio: $$RDR=\frac{P(top)}{P(top/A=0)}$$ Risk Increase Ratio: $$RIR=\frac{P(top/A=1)}{P(top)}$$	N	66	Y

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
			Birnbaum Importance: $BB=P(top\|A=1)-P(top\|A=0)$			
			Risk Reduction Interval: $RRI=P(top)-P(top\|A=0)$			
			Risk Increase Interval: $RII=P(top\|A=1)-P(top)$			
Importance Measures	Selection techniques Select basic events to show their importance, this selection is accomplished through various selection techniques	Matching basic events with various criteria	Matching criteria follows: Primary name Group name System Train Type Failure mode Location Initiating event Attributes.	N	67	N
		Exclude basic events that match various criteria	Matching criteria follows: Primary name Group name System Train Type Failure mode Location Initiating event Attributes.	N	68	N
		Complement	Basic events that were not included or marked are now the selected events.	N	69	N
		Reset	Make all events selected.	N	70	N
		View events	Basic events that are marked (*) will be selected.	N	71	N

Table 1. (continued).

Major Function	Vital Feature	Subfeatures	Criteria/description	Boundary test (Y/N)	5.0 Vital feature number	Test performed (Y/N)
Report on database	Results report		Determine if the Report on Data Base function gives the same results as the Display Results function does for Fault Tree Analysis, Event Tree Analysis, etc. Verify that current and base case reports are appropriately identified and correct.	N	72	N
Seismic Analysis			As a task under W6241 Plant Database Development for SAPHIRE, the Surry seismic analysis will be loaded into SAPHIRE. Determine if the SAPHIRE results are comparable to the NUREG-1150 results as documented in Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events (NUREG/CR-4550).	N	73	Y

Once the features and subfeatures to be tested had been identified, a software test record was developed for each test. The software test record specifies the input, execution conditions, and expected results for the test. As each test was performed, the actual results were also documented on the software test record.

Anomaly reports were prepared for all discrepancies identified between the way SAPHIRE should operate and actual operation. Discrepancies were categorized as either critical or noncritical. A discrepancy is categorized as critical if the results are incorrect or the potential exists for the results to be misleading. A discrepancy is categorized as noncritical if the results are correct but the option is difficult to use.

5.2 User Interviews

Once the initial V&V test procedures had been developed, the next step was to evaluate how complete these initial test procedures were. The objective of this task was to determine if the initial V&V test procedures adequately covered problem areas that users had encountered in their use of prerelease versions of SAPHIRE 5.0. To accomplish this task, interviews were conducted with users who had both rigorous prior experience with earlier IRRAS and SAPHIRE versions and were thus well acquainted with the code, and were using SAPHIRE 5.0 on a current project. Next, a summary of all problems and difficulties was created and compared against the initial vital and nonvital features test procedures to determine if additional tests needed to be added to test the problems disclosed by the interview process.

5.2.1 Methodology

There were eight interviews conducted, seven of which were from individuals at the Idaho National Engineering Laboratory (INEL) and one from Sandia National Laboratories. The limited industry diversity in the interview group reflects the limited distribution of prerelease or beta test versions of SAPHIRE 5.0. The interviews were conducted in person or by phone, and all followed the format shown in Figure 1. The form consists of three sections. The first section is a general information section that includes the individual's name, organization, and computer type. The second section is a SAPHIRE 5.0 user section that includes the SAPHIRE 5.0 version being used (i.e., the release date) and the length of time using that version of SAPHIRE 5.0. The third section includes the features most often used, a description of undocumented features encountered, a description of problems or difficulties encountered, the system or functionality involved, and any action taken to correct the problem. The results from the interviews were accumulated and compared to the initial vital and nonvital features test criteria.

5.2.2 Results

A summary of the problems and difficulties that users identified are shown in Table 2. This accumulated problem list contains 12 items that are considered to fall into the program error category, 3 items in the undocumented features category, and 7 items in the user interface category. The items that fall into the program error and undocumented feature categories are addressed below. While the items that fall into the user interface category are not considered to impact the vital and nonvital features test procedures, they are also listed in Table 2 for completeness, but are not addressed in the following discussion.

User name:
Organization:
Computer type:

SAPHIRE 5.0 release date:
Length of time using SAPHIRE 5.0:

Describe features most often used:

Describe any undocumented features encountered:

Describe problems or difficulties encountered:

What system or functionality was involved:

Describe any action taken to correct the problem:

Figure 1. User interview form.

Table 2. Accumulated problems list.

Program Errors

- Entering a one gate logic file resulted in replacement of the gate and event with the same event name.
- Trees with >9,999 gates trashed (corrupted) the database and completed the run not showing the user that there was a problem.
- Generating sequences with success branches caused a failure in generating sequence cut sets. The code ignored the problem and eventually showed a "Fatal disk error—scratch drive probably full" message.
- Fault tree logic is correct on the screen but logic created from the tree is incorrect. Renamed a basic event but the basic event was named after a previous gate. A problem in the conversion from graphics to logic file.
- Uncertainty analysis—quantification of sequences—results in code obtaining a negative value in the min cut upper bound. Sample distribution types include lognormal, constants, and uniform distributions.
- One end state when requantified was zero for the minimum cut set but each cut set in the end state had the proper value.
- Report function does not properly display results unless the system or sequence you are interested in is quantified.
- Using the change event tree function gets a "reference map bad" message and forces a rebuild on the database.
- The logic editor automatically changes the gate names in one tree when other gate names in another tree are changed creating a duplicate gate name error in a sequence.
- When modifying an event tree by deleting and adding a sequence, the code will duplicate some branches. This causes a duplication in sequence names which in turn causes incorrect frequency values in the Link Editor.
- When comparing two unique cases (i.e., two unique initiating events) in the sensitivity analysis by setting the initiating events to false for the set not being used everything worked all right. However, when running importance on these results the analysis nulled all values out (perhaps a divide by zero error or incorrect fix).
- The current case and base case were mixed up thus making the reports incorrect.

Table 2. (continued).

Undocumented Features

- Seismic, end state uncertainty, prune cut sets (basically anything not in the 4.0 version of the documentation because the new documentation is not out yet).
- In the Utility menu the options are associated with letters L, D, X, A, and R. However, if you choose M, a Modify Version date screen is available. This feature is not documented.
- The S feature in one of the menus takes you to SARA.

User Interface

- When choosing a submenu option, the default menu option is set to exit. This brings you back to the main menu.
- Displays sequences in a list that have no cut sets.
- It is often necessary to recover the database. No clear indication of when this is appropriate or not.
- Enter and ESC keystrokes are inconsistently used.
- In the help functions, the selected item is chosen by pressing ESC or ENTER. There is no cancel command in the help function.
- When modifying through the fault tree logic editor, changes were not automatically made to the graphics portion of the code. To get the logic changes incorporated into the graphics requires that the Alpha to Graphics operation be performed. User felt that by modifying the logic, the graphics should be changed by implication.
- Prune cut sets does not allow multiple rules for cut sets. They must be used individually. The prune command should allow users to apply all rules to all sequences or system cut sets.

Program Errors. Six of the twelve items in the program error category were identified as not being covered in the initial vital features test procedures. Of these six, three are associated with the Fault Tree Logic Editor, two are associated with the Report function, and one is associated with the Event Tree Analysis function. Therefore, the following tests were added to the vital features test procedure to compile a more thorough V&V effort.

Although there were several procedures to test the fault tree analysis function, there were initially no procedures to test the functionality of the logic editor and the fault tree graphics editor. Problems with the logic editor are represented by Items 1, 4, and 9 on the accumulated problem list. An additional test was added to the vital features test procedure to test the logic editor by: (a) entering one logic gate and checking the gate and event name, (b) modifying gate names in one tree and checking that gate names in other trees have not been modified and a duplicate gate error message is not displayed, and (c) comparing the results of the fault tree graphic editor with the logic editor.

It was also determined that the initial test procedures did not include tests of the Report generation function. A test was developed to determine if the generated report properly displays results of unquantified systems or sequences, and that the current and base case reports are not mixed-up.

The initial event tree test procedures were also modified to test adding or deleting a sequence and then checking for duplicate sequence names and incorrect frequency values in the Link Editor.

Undocumented Features. It should be noted that the user interviews were conducted before documentation had been released for SAPHIRE 5.0. One of the three items in the undocumented features category was identified as not being covered in the initial test procedures. The Utility menu includes an option to modify the version date by choosing an M instead of the possible L, D, X, A, R menu options. If this option is available to the user, it should be tested; otherwise, it should be removed from the program.

5.2.3 Conclusions

This task was found to be a very useful means for evaluating how complete the vital and nonvital test procedures were. There were some problems and difficulties encountered by users that were identified as not being included in the initial V&V test procedures. Additional tests have been added to the V&V test procedures to compile a more thorough V&V effort. For future versions of SAPHIRE, it is recommended that the code be released to a wider prerelease and beta test audience so that more users can be contacted.

6. VITAL FEATURES TEST RESULTS

Except for the seismic analysis function, all vital features were tested on SAPHIRE Version 5.0 dated July 20, 1994. The seismic analysis function was tested on SAPHIRE 5.02. The majority of the results were determined to be acceptable. The critical discrepancies that were identified between the expected operation and the actual operation of SAPHIRE are discussed in greater detail in the following sections. Discrepancies were categorized as either critical or noncritical. A discrepancy is categorized as critical if the results are incorrect or the potential exists for the results to be misleading. A discrepancy is categorized as noncritical if the results are correct but the option is difficult to use. Both critical and noncritical discrepancies were identified.

The software test records (STRs) for all tests that identified a discrepancy are presented in Appendix B. Because the STRs include a discussion of the discrepancy, the corresponding anomaly reports have not been included in Appendix B. The complete set of vital features STRs and anomaly reports are kept on file and can be requested from any of the authors.

6.1 Critical Discrepancies

There were several discrepancies that were categorized as critical. These discrepancies occurred in the functional areas of cut set generation, event probability cutoff, uncertainty analysis, and the database conversion routine.

6.1.1 Cut Set Generation for a Large Fault Tree—STR-03

The purpose of this test was to determine if the correct cut set are generated when a large fault tree is analyzed. There were two tests performed. For the first test five fault trees from the LaSalle database were combined to produce a new tree that consisted of 683 gates. Cut sets for this tree were generated using both SAPHIRE and CAFTA. The results from this test were that the SAPHIRE and CAFTA cut sets were the same.

However, another test was performed for the purpose of verifying the SAPHIRE limit on the maximum number of gates that it can process (i.e., 9,999 gates). For this test a tree was constructed by "OR"-ing one gate and two events under each gate with the final gate having just the two events. The two events used with each gate are identical except for the first gate, where an extra unique event was added. The tree was broken into smaller parts for easier loading into SAPHIRE. The final tree was constructed by combining the broken parts of the tree with an AND gate. The acceptance criteria were that the cut sets should consist of two first order cut sets representing the two events that are the same for each gate plus a cut set containing the product of the unique event added to each top gate. The actual results were that there were two second order cut sets produced that contained the product of one of the two events combined with an event titled "Undeveloped Gate." By checking the logic under the REPORT-SYSTEM-LOGIC menu, it was found that SAPHIRE collapsed each branch into temporary gates that had the same names and then combined the temporary gates with the same name in an AND gate. Combining the same gates in the AND gate caused SAPHIRE to produce the erroneous "Undeveloped Gate."

This discrepancy was categorized as critical because the results are incorrect. Even though it would not be common for a fault tree to be composed entirely of gates with the same events connected by OR gates, especially not one this size, it cannot be ruled out entirely. The results would also be incorrect for a small tree of only two or three levels that consist of gates with the same events.

In performing this test, it was also identified that it would be helpful to the user if a warning message was produced that states that the 9,999-gate limit has been reached and no further gates should be added. It also appears that there is a limit of 2,500 gates that can be read in. This limit was not explicitly tested because it is not a vital feature, but was the reason the 9,999-gate tree was broken into smaller parts.

6.1.2 Event Probability Cutoff Feature—STR-22

The purpose of one of the subtests for STR-22 was to determine if the cut sets are correct when the event probability cutoff feature is used. This test was performed by substituting the quantified fault tree cut sets into the event tree top events and then performing the test again by substituting the fault tree logic into the event tree top events. The acceptance criteria were that the cut sets using the quantified fault tree substitution method would be the same as the cut set using the fault tree logic substitution method. The actual results were that the quantified fault tree substitution method produced 769 cut sets while the fault tree logic substitution method produced 14 cut sets. The first 13 cut sets for each method were identical; however, the fourteenth cut set for the quantified fault tree substitution method did not match the fourteenth cut set for the fault tree logic substitution method. Instead, the fourteenth cut set for the fault tree logic substitution method matched the fifteenth cut set for the quantified fault tree substitution method. Furthermore, the fourteenth cut set for the quantified fault tree substitution method was a legitimate cut set. Therefore, one or both of the results are erroneous and the event probability cutoff feature does not work.

Even though this feature is not commonly used, this discrepancy was categorized as critical because the results are incorrect.

6.1.3 Uncertainty Analysis—Various STRs

The uncertainty analysis was checked for both the Monte Carlo and Latin Hypercube sampling. For both of these sampling techniques, single event analysis (i.e., a single basic event) and multiple event analysis (i.e., a group of basic events, such as system or sequence cut sets) were investigated. All SAPHIRE available distributions were checked. These distributions are lognormal, normal, beta, chi-square, exponential, uniform, histogram, gamma, and maximum entropy.

The discrepancies identified for the uncertainty analysis function are shown in Table 3. A majority of the discrepancies occurred due to the uncertainty not being correctly calculated for those cases where the value of 1.0 is an acceptable input parameter. For a normal distribution using either the Monte Carlo or the Latin Hypercube sampling techniques, the code also will not calculate the uncertainty if the mean value is set equal to zero even though a mean frequency value of zero is

Table 3. Uncertainty analysis function discrepancies.

STR	Description	Uncertainty not correctly calculated for an acceptable value of 1	Other discrepancies identified
45-01A	Monte Carlo-lognormal-basic event	Yes	No
45-02A	Monte Carlo-normal-basic event	No	For one case, a normal distribution with a mean value = 0, a standard deviation = 1, and the basic event indicated as an initiating event, a message at the bottom of the screen indicates that a frequency value = 0 implies no uncertainty and any uncertainty data will be ignored. A mean frequency value = 0 is acceptable for a normal distribution. For a second case, if the basic event is not indicated as an initiating event, the generated distribution was truncated at 1.0 but was not truncated at 0.0 (negative probability values allowed).
45-03A	Monte Carlo-beta-basic event	Yes	Typical sample values are treated as zero. The code will allow the user to input incorrect values for this distribution.
45-04A	Monte Carlo-chi square-basic event	Yes	No
45-04B	Monte Carlo-chi square-different sample sizes-basic event	Yes	No
45-05A	Monte Carlo-exponential-basic event	Yes	No

Table 3. (continued).

STR	Description	Uncertainty not correctly calculated for an acceptable value of 1	Other discrepancies identified
45-06A	Monte Carlo-uniform-basic event	Yes	For a uniform distribution with a mean = 0 and low value = −1, the code does not allow the distribution to have negative values.
45-07A	Monte Carlo-histogram-basic event	Yes	The code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram being evaluated should be determined by the code and utilized; otherwise, incorrect point estimates will be calculated.
45-08A	Monte Carlo-gamma-basic event	Yes	No
45-09A	Monte Carlo-maximum entropy-basic event	No	Case A. For the maximum entropy distribution with a mean = 5E−1, lower end = 0.00, and upper end = 1.00, the results returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.5. The standard deviation turned out to be 0 instead of the correct value of 0.2887. Case E. For the maximum entropy distribution with a mean = 9.999E−01, lower end = 0.00, and upper end = 1.00, the results returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.999. The standard dviation turned out to be 0 while the skewness and kurtosis were 1.0E+38.
46-01A	Latin Hypercube (LH)-lognormal-basic event	Yes	No

Table 3. (continued).

STR	Description	Uncertainty not correctly calculated for an acceptable value of 1	Other discrepancies identified
46-02A	LH-normal-basic event	No	For one case, a normal distribution with a mean value = 0, a standard deviation = 1, and the basic event indicated as an initiating event, a message at the bottom of the screen indicates that a frequency value = 0 implies no uncertainty and any uncertinty data will be ignored. A mean frequency value = 0 is acceptable for a normal distribution. For a second case, if the basic event is not indicated as an initiating event, the generated distribution was truncated at 1.0 but was not truncated at 0.0 (negative probability values allowed).
46-03A	LH-beta-basic event	Yes	Cases F and G. For beta distributions with mean values = 1E−6 and uncertainty values = 2.0 and 8E+5, respectively, the generated results for these two cases are incorrect. Case J. The code will allow the user to input incorrect values for this distribution.
46-04A	LH-chi square-basic event	Yes	No
46-04B	LH-chi square-basic event	Yes	No
46-05A	LH-exponential-basic event	Yes	No
46--6A	LH-uniform-basic event	Yes	For a uniform distribution with mean = 0 and low value = −1, the code does not allow the distribution to have negative values.

Table 3. (continued).

STR	Description	Uncertainty not correctly calculated for an acceptable value of 1	Other discrepancies identified
46-07A	LH-histogram-basic event	Yes	The code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram being evaluated should be determined by the code and used; otherwise, incorrect point estimates will be calculated.
46-08A	LH-gamma-basic event	Yes	No
46-09A	LH-maximum entropy	No	Case A. For the maximum entropy distribution with a mean = 5E−1, lower end = 0.00, and upper end = 1.00, the results returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.2887. Case E. For the maximum entropy distribution with a mean = 9.999E−01, lower end = 0.00, and upper end = 1.00, the results returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.999. The standard deviation turned out to be 0 while the skewness and kurtosis were 1.0E+38.

acceptable for a normal distribution. For the beta distribution using the Monte Carlo sampling technique, sample values within the range of those typically used in a PRA are being treated as if the values were zero. For the beta distribution using either sampling technique, the code will allow the user to input incorrect values. For the histogram distribution using either sampling technique, the code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram should be determined by the code and used; otherwise, incorrect point estimates will be calculated. For input values that would be expected in a PRA, incorrect results were obtained for some test cases associated with the maximum entropy distribution using either sampling technique and for the beta distribution using the Latin Hypercube sampling technique. The code is also inconsistent with respect to the treatment of negative values. Negative values are allowed for some distributions but not for others.

6.1.4 Database Conversion Routine—STR-90

A test of the database conversion routine was added as an unplanned test when it was noted that the number of cut sets for another SAPHIRE 5.0 test were not comparable to the number of cut sets when the test had been performed for SAPHIRE 4.0 even though the test had been successfully run under SAPHIRE 4.0. The database conversion routine became suspect when the correct number of cut sets were obtained after the database was extracted from SAPHIRE 4.0 as "flat files" and uploaded into SAPHIRE 5.0. Therefore, the purpose of this test was to determine the validity of the database conversion routine for SAPHIRE 5.0. This test will be added as a planned test for all future V&V efforts of SAPHIRE.

The test was performed by converting a SAPHIRE 4.0 database from the 4.0 database format to the 5.0 database format. Reports listing cut sets, frequencies, and events ranked in order of importance were extracted from SAPHIRE 5.0 and compared to the same reports for SAPHIRE 4.0. The results did not identically match. This discrepancy was categorized as critical because the results were incorrect and because use of the database conversion routine is more of an established procedure than uploading flat files.

6.2 Other Comments on Test Results

Even though no discrepancies were found with the seismic analysis feature, a discussion of the test and results was considered to be warranted because it is a major new function of SAPHIRE. The seismic analysis feature was tested on SAPHIRE 5.02. The test for the seismic analysis module of SAPHIRE consisted of the conversion of the seismic analysis results from NUREG/CR-4550,[4] Vol. 3, Rev. 1, Part 3, "Analysis of Core Damage Frequency: Surry Power Station, Unit 1 External Event" to a working SAPHIRE seismic database. Seismic cut sets and uncertainty results generated from SAPHIRE were compared to the results obtained in the NUREG/CR-4550 PRA analysis. Many of the seismic analysis results generated by SAPHIRE were also checked by hand calculations.

The SAPHIRE seismic module is designed to function directly from the internal database assumed to be already available to the user. Specifically, random failure composed system models, accident sequence progression, and initiating events have all been defined and developed for the engineered system of interest (i.e., for this test, the Surry Power Station, Unit 1). However, for the Surry external event analysis, the analysis was done with separate seismic system fault trees and initiating events. Therefore, the SAPHIRE database also contains separate seismic system fault trees

developed from the cut sets obtained from NUREG/CR-4550 PRA analysis and does not generate seismic analysis directly from the internal database.

To perform seismic analysis, SAPHIRE must modify the internal "random" basic events to include seismic-induced failures. This is done by defining transformations of the random failures modeled in the internal events analysis into seismic failures. The transformation process was checked using visual confirmation. All seismic susceptible internal basic events were confirmed to be transformed into the new seismic basic event.

The seismic systems created in SAPHIRE for the Surry analysis were obtained from the cut sets generated in the external event analysis of NUREG/CR-4550. The new seismic fault trees were created using the internal basic event names and probabilities. MAR-D system logic files were created and loaded into the Surry database from the NUREG/CR-4550 seismic cut sets. The test for the SAPHIRE seismic system cut sets involved the regeneration of the seismic cut sets from the logic files. No discrepancies were found in the generation of seismic system cut sets.

Sequence cut sets for seismic analysis were generated from SAPHIRE using seismic event trees. These seismic event trees were developed from the Boolean logic and the dominant seismic sequences given in NUREG/CR-4550. Seismic sequence cut sets were checked and confirmed by hand and with those produced in NUREG/CR-4550. No discrepancies were found in the generation of seismic sequence cut sets.

Seismic quantification for the Surry database was performed using a hazard curve or seismic histogram. The hazard curve is input through SAPHIRE and represents a range of possible earthquakes in the form of an exceedence curve. Quantification is performed at each of the earthquake ground accelerations (earthquake strengths) that was input. Hand checks confirmed that SAPHIRE does quantify using the proper values from the hazard curve.

The uncertainty algorithms and parameters used in NUREG/CR-4550 for the seismic analysis were used to check the seismic uncertainty results of SAPHIRE. Hand calculations confirmed that no discrepancies were determined using the seismic uncertainty analysis of SAPHIRE. It should be noted that even though the hand calculations confirmed that there are no discrepancies, in some cases final core damage frequency results were significantly different when compared to the NUREG/CR-4550 results. The direct cause of the differences are unknown but several factors could contribute. For example, the NUREG/CR-4550 mentions truncation and sequence screening but the analysis description is unclear regarding what truncation and at what level it took place. In addition, many of the data analysis results from the NUREG/CR-4550 are vague in the analysis descriptions and could not readily be regenerated by hand.

7. SOFTWARE TESTING OF NONVITAL FEATURES

SAPHIRE features were evaluated and either designated as vital or nonvital features. Nonvital features are those features whose malfunction or lack of function will not preclude obtaining the final correct results from SAPHIRE. The nonvital features are fault tree construction, event tree construction, database manipulation, histograms, results display, and general capabilities. Lack of function of a nonvital feature can add many additional hours to an analysis because the work that would have been performed by that feature will have to be performed manually or through an alternate method.

Nonvital features were tested for functionality. The first step was to prepare a test design specification. The purpose of the test design specification was to identify and summarize the nonvital features to be tested. The next step was to prepare a test procedure. The purpose of the test procedure was to identify the success criteria that the feature must meet. The final step was to apply the test procedures to SAPHIRE. Software test records were not generated for the nonvital features. Instead, the test procedure was followed and items were checked off as they were completed. All discrepancies identified with the nonvital features were documented on anomaly reports.

The nonvital features testing was performed on the October 21, 1993, beta version of SAPHIRE 5.0. Three discrepancies were identified and all were categorized as noncritical. One of the discrepancies involved not being able to change the font within the create event tree feature of IRRAS. The second discrepancy involved the "transfer back" function of file manipulation in FEP. The final discrepancy involved not being able to change the "name size" in FEP.

The three anomaly reports are presented in Appendix C along with the applicable procedures. The complete nonvital test design specification and test procedure documents for SAPHIRE 5.0 are kept on file and can be requested from any of the authors.

8. CONCLUSIONS AND RECOMMENDATIONS

Section 8.1 discusses the V&V findings that are directly applicable to SAPHIRE and provides recommendations for these findings. Section 8.2 discusses the findings from the V&V process that were used and provides recommendations for improving this process.

8.1 Conclusions and Recommendations from SAPHIRE V&V

The following summarizes the major findings from the review of the code development control procedures and the vital features testing. As appropriate, recommendations are provided.

8.1.1 Code Development Control Procedures

The NRC *Software Quality Assurance Program and Guidelines* identifies three software levels that reflect the intended purpose and use of the software including whether or not it will be used in making safety decisions.[3] The extent of the V&V and the necessary code control procedures can be better established if the software level is known. It is recommended that consideration be given to establishing the software level for SAPHIRE so that the development and V&V activities are commensurate with the software level.

For SAPHIRE 4.0, the requirements were documented in *Integrated Reliability and Risk Analysis System (IRRAS), Version 4.0, Functional Requirements Document* (Draft).[b] However, for SAPHIRE 5.0, the requirements were not documented. Because no documentation of the requirements was prepared, it was not possible to review the requirements for completeness, correctness, consistency, or quality. The lack of documented requirements also limits the ability of the sponsor/user community to perform thorough acceptance testing or to formally determine if the project has been successfully completed. For future versions of SAPHIRE it is recommended that detailed software requirements be developed and documented. Even though it was identified in the V&V of SAPHIRE 4.0 that some improvements were needed to the IRRAS 4.0 functional requirements document, it did serve as a first step in the right direction of establishing a software development library as recommended in NUREG/BR-0167.[3]

It is also difficult to derive a complete, consistent, and correct set of requirements without sponsor/user involvement in reviews of the requirements. For major modifications and enhancements, it is recommended that consideration be given to conducting a requirements gathering meeting and at later dates holding requirements and design reviews. The purpose of the meetings is user input to help clarify and finalize the requirements. The requirements gathering meeting can provide the development team with an excellent opportunity to get the users to describe in detail all of the steps that a PRA analyst goes through when performing a certain PRA process. The users should understand that they are not there to perform the coding—they are there to describe the PRA process in sufficient detail so that the development team can translate the process into coding that will perform the PRA process in a more automated manner. The descriptions of the PRA process form the basis for the requirements. The later requirements and design reviews provide the development team with the opportunity to check their interpretation of the process.

---

b. By S. Ted Wood and Kenneth D. Russell, EG&G Idaho, Inc., Idaho Falls, Idaho, December 1991.

It is recommended that consideration be given to coordinating the development and V&V projects so that there is a more comprehensive testing program that occurs over the entire life cycle. The testing program should address the type of testing (unit, integration, system, and acceptance) that will be performed and the test documentation requirements.

Generally, acceptance testing is performed before the sponsor accepts the code for production purposes. It is recommended that consideration be given to coordinating the development and V&V projects so that acceptance testing is performed before the code is released to the Energy Science and Technology Software Center; however, it should be recognized that this can result in the need for iterative acceptance testing. To limit the number of iterations of acceptance testing, it is recommended that once the development team has completed its internal testing and beta testing and is satisfied that the code is ready for production use, it be formally transferred to the V&V team for acceptance testing.

It is recommended that consideration be given to not using beta test versions on projects that should be using production software. Not only is the credibility of the SAPHIRE code being affected by this practice, but the budgets and schedules of these other projects are also being impacted.

8.1.2 Vital Features Test Results

Except for the seismic analysis function, all vital feature were tested on SAPHIRE Version 5.0 dated July 20, 1994. The seismic analysis function was tested on SAPHIRE Version 5.02. Overall, the majority of the test results were determined to be acceptable. Critical discrepancies were identified in several areas and were provided to the code developers for modifications to SAPHIRE 5.0. The correction of the identified discrepancies was not retested. The critical discrepancies are discussed in detail in Section 6.1. Briefly, the critical discrepancies identified are as follows:

Cut Set Generation. To verify the limit on the maximum number of gates that SAPHIRE can process (9,999 gates), a tree was constructed by "OR"-ing one gate and two events under each gate with the final gate having just the two events. The code was able to process 9,999 gates; however, the cut sets were incorrect. By checking the SAPHIRE generated logic, it was determined that SAPHIRE collapsed each branch into temporary gates that had the same names and then combined the temporary gates with the same name in an AND gate. Combining the same gates in the AND gate caused SAPHIRE to produce an erroneous "Undeveloped Gate." Even though it would not be common for a fault tree to be composed entirely of gates with the same events connected by OR gates, especially not one this size, the same incorrect results would be obtained for a small tree of only two or three levels. This discrepancy was provided to the code developers and was corrected with the release of SAPHIRE 5.01.

Event Probability Cutoff Feature. When the event probability cutoff feature was tested, it was found that the results were different between the fault tree logic case and the quantified fault tree cut set case. This discrepancy is being evaluated by the code developers and will be resolved with a later release of SAPHIRE.

Uncertainty Analysis Function. There were several discrepancies identified for the uncertainty analysis function. A majority of the discrepancies occurred due to the uncertainty not being correctly calculated for those cases where the value of 1.0 is an acceptable input parameter. For a normal

distribution using either the Monte Carlo or the Latin Hypercube sampling techniques, the code also will not calculate the uncertainty if the mean value is set equal to zero even though a mean frequency value of zero is acceptable for a normal distribution. For the beta distribution using the Monte Carlo sampling technique, sample values within the range of those typically used in a PRA are being treated as if the values were zero. For the beta distribution using either sampling technique, the code will allow the user to input incorrect values. For the histogram distribution using either sampling technique, the code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram should be determined by the code and used; otherwise, incorrect point estimates will be calculated. For input values that would be expected in a PRA, incorrect results were obtained for some test cases associated with the maximum entropy distribution using either sampling technique and for the beta distribution using the Latin Hypercube sampling technique. The code is also inconsistent with respect to the treatment of negative values. Negative values are allowed for some distributions, but not for others. These discrepancies were provided to the code developers. The discrepancies for those cases where the value of 1.0 is an acceptable input parameter was corrected with the release of SAPHIRE 5.03. The remaining discrepancies were corrected with the release of SAPHIRE 5.05.

Database Conversion Routine. A test of the database conversion routine was added as an unplanned test when it was noted that the number of cut sets for another SAPHIRE 5.0 test were not comparable to the number of cut sets when the test had been performed for SAPHIRE 4.0 even though the test had been successfully run under SAPHIRE 4.0. The database conversion routine became suspect when the correct number of cut sets were obtained after the database was extracted from SAPHIRE 4.0 as "flat files" and uploaded into SAPHIRE 5.0. This discrepancy was provided to the code developers and was corrected with the release of SAPHIRE 5.03.

8.2 Conclusions from the V&V Process

By performing the V&V on the same version of the code, better assurance can be given that there are no unwanted or incorrect interactions between related and dependent functions within the code. When different code versions are used, no assurance of this type can be given. It was originally intended that all vital features would be tested on the same version of the code; however, during the use of SAPHIRE 5.0 to load the Surry seismic analysis it was found that some modifications to the seismic analysis function were necessary. As a result, the seismic analysis function was tested using SAPHIRE Version 5.02. Many of the recommendations made in Section 8.1.1 would help ensure that the V&V is performed on one version of the code.

9. REFERENCES

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

2. IEEE Standard 1012-1986, "IEEE Standard for Software Verification and Validation Plans," Institute of Electrical and Electronic Engineers.

3. *Software Quality Assurance Program and Guidelines*, NUREG/BR-0167, February 1993.

4. R. C. Bertucio and J. A. Julius, *Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events*, NUREG/CR-4550, April 1990.

Appendix A

Verification and Validation Plan for SAPHIRE 5.0

Appendix A

Verification and Validation Plan for SAPHIRE 5.0

This document defines the software verification and validation (V&V) plan to be used for an evaluation of the System Analysis Programs for Hands-on Integrated Reliability Evaluation (SAPHIRE) 5.0 code suite. This V&V plan is based on the IEEE Standard 1012-1986. However, because this V&V is being performed after the beta testing and after SAPHIRE 5.0 has been released to the Energy Science and Technology Software Center, not all the V&V activities outlined in the IEEE Standard 1012-1986 for each life-cycle phase will be performed. All life-cycle phases that will not be applied will be so indicated. In addition, the V&V plan for SAPHIRE 5.0 is based on the V&V plan that was originally developed for the V&V of SAPHIRE 4.0 (see NUREG/CR-6145). The V&V plan for SAPHIRE 4.0 has been revised to be applicable to SAPHIRE 5.0 and to account for lessons learned during the SAPHIRE 4.0 V&V effort.

SAPHIRE is a set of four computer programs that the Nuclear Regulatory Commission (NRC) has developed for the performance of probabilistic risk assessments (PRAs). These programs allow an analyst to perform many of the functions necessary to create, quantify, and evaluate the risk associated with a facility or process being analyzed. The programs included in this set are Integrated Reliability and Risk Analysis System (IRRAS), System Analysis and Risk Assessment (SARA), Models and Results Database (MAR-D), and Fault Tree/Event Tree/Piping and Instrumentation Diagram (FEP) graphical editor. These programs are "hands-on" in the sense that they execute on an IBM-compatible PC, and they are integrated in the sense that they can communicate with one another through the MAR-D database.

A-1. PURPOSE

A documented V&V of SAPHIRE 5.0 is necessary to demonstrate that the software is adequate for expected production applications. This V&V plan is proposed for the overall purpose of completing a documented V&V to determine if the results produced by SAPHIRE 5.0 are correct, and if not, to identify any areas requiring correction.

A-2. REFERENCED DOCUMENTS

This V&V plan shall comply with the following documents:

1. IEEE Standard 1012-1986, "IEEE Standard for Software Verification and Validation Plans."

2. ANSI/ASME NQA-1 Supplement 11S-2, "Supplementary Requirements for Computer Program Testing."

Documents referenced by this V&V plan are:

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

2. ANSI/ANS-10.4-1987, "Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry."

3. ANSI/IEEE Standard 829-1983, "Test Documentation."

4. R. C. Bertucio and J. A. Julius, *Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events*, NUREG/CR-4550, April 1990.

Supporting documents required to supplement or implement this V&V plan are:

1. Code development control procedures such as the project plan, software requirements specification for SAPHIRE 5.0, design specification for SAPHIRE 5.0 and configuration management plan.

2. The system environment specification of the SAPHIRE 5.0 version. IBM-PC/XT/AT PS2 or 100% compatible with 640 K main memory, DOS 3.3 or later, 20 M hard disk (minimum), and math co-processor (optional).

3. The SAPHIRE 5.0 Users Manual.

A-3. DEFINITIONS

In the context of this V&V plan, the following definitions apply:

Anomaly. Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents. A critical anomaly is a failed vital feature and must be resolved before the V&V effort can proceed.

Beta testing. Testing and notification of anomalies provided by software users.

Vital feature. A vital feature is essential for completing the PRA analysis. If this feature failed, it would be impossible to get final PRA results from SAPHIRE 5.0.

Life-cycle phase. Any period of time during software development or operation that may be characterized by a primary type of activity (such as design and testing) that is being conducted.

Nonvital feature. Nonvital features are those features whose malfunction or lack of function will not preclude getting the final results from SAPHIRE 5.0. Lack of function of a nonvital feature can add many additional hours, days, or months to an analysis because the work that has been done by that feature will have to be done by hand.

Software requirements specification. Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.

Test case specification. Documentation specifying inputs, predicted results, testing procedures, and a set of execution conditions for a test item.

Test design specification. Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests.

Validation. The process of evaluating software at the end of the software development process to ensure compliance with software requirements obtained from the software requirements specification.

Verification. The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase.

Vital Feature. Vital features are those features that (a) affect the results of a PRA (core damage frequency, top contributors, etc.) and (b) are essential for completing a PRA analysis, such as fault tree analysis, event tree analysis, uncertainty analysis, change set features, and importance measures.

A-4. VERIFICATION AND VALIDATION OVERVIEW

A-4.1 Organization

To be completed by the principal investigator (PI).

A-4.2 Master Schedule

To be completed by the PI.

A-4.3 Resources Summary

The resources, such as staffing, facilities, tools, finances, and special procedural requirements, shall be defined by the PI. Emphasis will be placed on the most important software elements with the scope of the V&V being balanced with cost and schedule constraints.

A-4.4 Responsibilities

The PI has responsibility for specifying and meeting budget, schedule, and V&V plan requirements. The PI will direct and oversee performance of the V&V.

A-4.5 Tools, Techniques, and Methodologies

Under the V&V of SAPHIRE 4.0, test cases were developed by experienced users to test vital and nonvital features of the code. For the V&V of SAPHIRE 5.0, experienced users have revised

and modified these tests and have developed additional tests so that the test cases are applicable to SAPHIRE 5.0 and lessons learned from the SAPHIRE 4.0 V&V effort have been incorporated as appropriate.

Where possible, test results will be compared to theoretically-based hand calculations and/or to results from the SAPHIRE 4.0 test cases. Where hand calculations are not practical, existing codes such as CAFTA and PC-SETS software will be used to provide computer-assisted results for comparison. Although neither CAFTA and PC-SETS have been verified and validated, it is deemed appropriate that correlation of results with SAPHIRE signifies correct operation of the software because the three codes use different coding techniques and methodologies.

A-5. LIFE-CYCLE VERIFICATION AND VALIDATION

A-5.1 Management of V&V

1. *Software Verification and Validation Plan Generation.* This document will serve as the software V&V plan, and can be revised by the PI as necessary.

2. *Baseline Change Assessment.* All V&V tests must be performed on one sub-version of the code. This V&V plan and the project budget and schedule do not accommodate an iterative cycle of producing anomaly reports, updating the code, and repeating the V&V test procedures.

3. *Management Review.* The PI will provide V&V progress reports, a draft V&V report, and the final V&V report to the Safety and Risk Evaluations unit manager and to the NRC technical manager.

4. *Review Support.* Other reviews may be requested by the PI such as independent reviews or external reviews. The PI should conduct reviews as required by the Quality Program Plan that the project is conducted under. Documentation of reviews will include as a minimum an internal correspondence summarizing what was reviewed, findings, and who performed the review.

5. *Applicable Life-Cycle Phases.* There have been several changes (modifications, enhancements, additions) that have been made to produce Version 5.0 of SAPHIRE. Per IEEE Standard 1012-1986 these changes should be treated as development activities and be verified and validated per the V&V tasks outlined for the concept, requirements, design, implementation, test, and installation and checkout phases. However, because this V&V is being performed after SAPHIRE 5.0 has been released to the Energy Science and Technology Software Center, the focus for this V&V effort has been placed on reviewing code development control procedures and performing a form of acceptance testing. Based on this, it is viewed that the V&V activities more logically fall under the operation and maintenance phase. As such, it should be recognized that this V&V effort does not represent a full V&V process as recommended by IEEE Standard 1012-1986.

   The V&V activities for this effort are provided in Section A-5.8.

A-5.2 Concept Phase V&V

Not applicable.

A-5.3 Requirements Phase V&V

Not applicable.

A-5.4 Design Phase V&V

Not applicable.

A-5.5 Implementation Phase V&V

Not applicable.

A-5.6 Test Phase V&V

Not applicable.

A-5.7 Installation and Checkout Phase V&V

Not applicable.

A-5.8 Operation and Maintenance Phase V&V

A.5.8.1 V&V Tasks

The following V&V tasks will be performed on SAPHIRE 5.0:

1. Evaluate the code development control procedures. The first step will be to determine what the minimum code development control procedures should be for any code no matter what development methodology is used (waterfall, cyclic, rapid prototyping, etc.). To do this, first identify the minimum V&V activities that are recommended to be performed from such existing standards as IEEE Standard 1012-1986. The inputs to performing these minimum V&V activities can then be viewed as representing the minimum code development control procedures. Some examples of V&V activities are evaluation of the software requirements document, evaluation of the design document, and review of configuration management procedures. The inputs, and thus the code development control procedures, would be development of the software specification and design documents and establishing and implementing configuration management procedures. The actual code development control procedures that were used for SAPHIRE 5.0 will then be evaluated against the minimum procedures that should be in place. Recommendations for improvements should be made as necessary.

2. Evaluate the completeness and correctness of the software requirements specification document. Generate an anomaly report for any discrepancies with the actual operation of the code.

3. Revise and modify the software tests that were originally developed for the SAPHIRE 4.0 V&V effort and develop additional tests as necessary to make all tests applicable to SAPHIRE 5.0 and to incorporate lessons learned from the SAPHIRE 4.0 V&V effort. Identify existing tests that stress the code boundaries (such as total number of gates processed) and develop additional boundary tests as necessary. Interview users to identify any undocumented features or features that they have had difficulty using with the beta version. Determine if the existing test cases cover these items, and if not, add new test cases. Software features that are designated as vital features are to be tested using the requirements stated in Section A-5.8.4. Software features designated as nonvital features are to be tested for functionality. The functionality tests will be recorded in a checklist fashion and any anomalies will be reported. Any untested features will be listed.

4. Throughout the V&V, evaluate completeness and correctness of user documentation particularly with regard to preventing code misuse. Generate an anomaly report for any discrepancies found (see item 5).

5. Tasks that are not planned include the following:

   a. Comprehensive inspection of source code; however, source code listings should be available for inspection.

   b. Comprehensive review of user documentation; however, anomalies detected should be reported.

   c. Review of the code development team's internal testing.

A-5.8.2 Software Features To Be Tested

Vital features to be tested are as follows:

1. *Fault Tree Analysis.* Test cases will be designed to test the fault tree cut set generation process, the quantifying process, and the capability to perform the analysis on a single fault tree or on multiple fault trees. The editing and modifying of fault tree cut sets are also tested in this process.

2. *Event Tree Analysis.* Test cases will be designed to test the event tree sequence generation process, sequence cut set generation process, quantifying process, and capability of performing the analysis on a single event tree or on multiple trees. The event tree rules and editing and modifying of event tree sequence cut sets are also tested in this process.

3. *Uncertainly Analysis.* Test cases will be designed to test the uncertainty analysis for the whole "family" (SAPHIRE designation for a entire PRA model) using both the Latin

Hypercube and the Monte Carlo simulation processes. The effect of sample size and seed number on the results will also be tested.

4. *Change Sets Feature.* Test cases will be designed to test the change sets feature and similar features used to perform sensitivity analyses.

5. *Cut Set Editor.* Test cases will be designed to test the cut set editor.

6. *Seismic Analysis.* The test for this feature will be performed as a task under W6241 Plant Database Development for SAPHIRE. This task covers the comparison of the results from loading the Surry seismic analysis into SAPHIRE against the NUREG-1150 results as documented in *Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events* (NUREG/CR-4550).

Nonvital features to be tested are as follows:

1. *Fault Tree Construction.* Tests will be performed to test fault tree construction including logic symbols and the functions in the Build Fault tree menu (e.g., alpha to graphics, pager, text fonts, scaling, printing capability).

2. *Event Tree Construction.* Tests will be performed to test the event tree construction including logic, sequences, plant damage states, and the functions in the Create Event tree menu (e.g., alpha to graphics, add/delete branch, add/delete top events, text fonts, scaling, printing capability).

3. *Database Manipulation.* The adding and deleting of basic events, event trees or fault trees, the base case update feature, the change sets update feature, etc., will be tested.

4. *Results Display.* The result displays and reports on screen and on hard copy will be tested.

5. *General Capabilities.* The general capabilities of SAPHIRE such as the online help features, the error messages, and the locate function (F5) will be tested.

A-5.8.3 Procedures for V&V

The V&V team under the direction of the PI are expected to follow the basic outline below:

1. Evaluate the code development control procedures. This includes reviewing the project plan for the development of the code, the requirements specification, the design specification, the configuration management plan, and nonconformance reporting and correction procedures.

2. Revise and modify as necessary the tests that were originally developed for the SAPHIRE 4.0 V&V effort to make these tests applicable to SAPHIRE 5.0. Develop additional tests to cover new SAPHIRE 5.0 features.

3. Execute the tests.

4. Report anomalies and progress.

5. For tasks that require a deviation from the V&V plan, create a deviation report.

6. Complete documentation of V&V and retain records as required in Section A-7.4.

A-5.8.4 Testing Requirements

For the vital features, tables that describe the feature or subfeature being tested and the criteria for performing the test will be revised and modified from those tables that were developed for the SAPHIRE 4.0 V&V effort to make the test applicable to SAPHIRE 5.0. New tests will be added to these tables to account for new SAPHIRE 5.0 features. The Nonvital Features Test Design Specification and Test Procedure will be revised as necessary to be applicable to SAPHIRE 5.0.

For each vital feature test case (or test segment) a software test record (STR) form will be completed and the following information will be provided:

- STR number for tracking
- Date of test
- Test performer
- Test description
- Computer system and operating environment
- Actual values versus expected values/acceptance criteria
- Input data files and listings
- Output data files and listings.

The STR will include any comments on adequacy and coverage of the test.

A numbering system for the test cases, STRs, anomaly reports, and so forth and a filing system for testing materials will be established. Configuration control of test cases, input and output files, and all project materials is required.

Subsequent revisions to SAPHIRE can use the tests generated in this V&V effort to test features that have not changed.

A-6. SOFTWARE VERIFICATION AND VALIDATION REPORTING

The documents to be created during the V&V effort are described below. Their distribution for review is described in Section A-5.1.

1. *V&V Progress Reports.* Lockheed Idaho Technologies Company will prepare a monthly letter status report in accordance with the current NRC Manual, Chapter 1102.

2. *Anomaly Report.* Anomaly reports shall be generated each time an anomaly is detected by the V&V effort. Each report shall include the control number for tracking purposes, description and location of the anomaly, its impact and cause (if known), and its criticality (as described in Section A-7.1 of the V&V plan). The draft and final V&V reports will contain the anomaly reports.

3. *Test Design Specification and Procedures.* For the vital features, tables that describe the feature or subfeature being tested and the criteria for performing the test will be revised and modified from those tables that were developed for the SAPHIRE 4.0 V&V effort to make the test applicable to SAPHIRE 5.0. New tests will be added to these tables to account for new SAPHIRE 5.0 features. The Nonvital Features Test Design Specification and Test Procedure will be revised as necessary. The draft and final V&V report will contain the vital features test description/criteria tables and the Nonvital Features Test Design Specification and Test Procedure.

4. *Test Summary Reports.* STRs will be prepared for each test. The contents of the STRs is described in Section A-5.8.4. The draft and final V&V report will contain the STRs (not including input and output file listings).

5. *V&V Final Report.* A V&V final report using NUREG/CR format will be generated. This report will summarize all life-cycle V&V tasks performed, all task results, test results, identified anomalies, assessment of overall software quality, and appropriate recommendations.

A-7. VERIFICATION AND VALIDATION ADMINISTRATIVE PROCEDURES

A-7.1 Anomaly Reporting and Resolution

Anomaly reports will be generated as anomalies are detected by the V&V team. Anomalies shall be ranked into one of two categories to assess the significance from a user's viewpoint as to the ability to produce accurate results:

1. *Critical.* The results are incorrect or the potential exists for the results to be misleading.

2. *Noncritical.* The results are correct but the feature/subfeature is difficult to use.

A-7.2 Task Iteration Policy

Even if critical anomalies are discovered, the current V&V plan and project budget and schedule do not accommodate an iterative process of modifying the code and repeating the V&V on a new subversion.

A-7.3 Deviation Policy

Deviations from the finalized V&V plan must be documented on a deviation form.

A-7.4 Control Procedures

All relevant materials (listings and files on magnetic media) including codes, reports, test plans, test case listings, and test input/output shall be retrievable and organized by a documented filing system. The records should be kept to aid future V&V needs as well as to provide quality records for the present project.

A-7.5 Standards, Practices, and Conventions

An appropriate Quality Program Plan (QPP) shall be created as designated by the PI.

SOFTWARE TEST RECORD - STR-

Date: _____
Originator: _____
Computer System: _____

Test Description

_____
_____
_____
_____

Acceptance Criteria

_____
_____
_____
_____

Results/Anomaly Description

_____
_____
_____
_____

Input Test Data File Name(s) and/or Output

_____
_____
_____
_____

Results File Name(s) and/or Output

_____
_____
_____
_____

ANOMALY REPORT - AR-

Date: _____
Originator: _____
Test Record Number: _____

Source Location

_____
_____
_____
_____

Description

_____
_____
_____
_____

Cause

_____
_____
_____
_____

Impact (Critical or Noncritical)

_____
_____
_____
_____

DEVIATION REPORT - DR-

Test Record No.: _____
Date: _____
Originator: _____

Software Test Number or Topic

_____
_____
_____
_____

Description

_____
_____
_____
_____

Justification

_____
_____
_____
_____

Anticipated Impact on V&V

_____
_____
_____
_____

Appendix B

Vital Features Software Test Records for Discrepancies

Appendix B

Vital Features Software Test Records for Discrepancies

There were several discrepancies identified when the SAPHIRE 5.0 vital features were tested. The software test records (STRs) for all tests that identified a discrepancy follow. Because the STRs includes a discussion of the identified discrepancy, the corresponding anomaly report is not included in this appendix. The complete set of vital features STRs and anomaly reports are kept on file and can be requested from any of the authors.

B-1. SOFTWARE TEST RECORD—STR-03-000-0894

Date: August 23, 1994
Originator: James L. Jones, Jr.
Computer System: ALR 486/33-MHz BV, 17-MB RAM, 340-MB HD, XGA Display, DOS 6.21

Test Description

Analyze a large fault tree to determine if the code correctly determines the appropriate cut sets. To accomplish this test, five fault trees are from the LaSalle Database (ADS, CDS, HPCS LPCI, and RCIC) are combined into a new tree named Test 3. The new tree is evaluated and the results are printed. Next the value block and tree information are downloaded into a sets input file and uploaded into CAFTA. The tree is evaluated in CAFTA, the output printed, and the results compared. The cutoff value for fault tree cut set generation used in this test is 1E−7, with a truncation of four events per cut set. The test tree contained 683 gates. The test for the maximum number of gates (9,999) is also performed. The test a tree was constructed by "OR"-ing one gate and two events under each gate with the final gate having just the two events. The two events used with each gate are identical except for the first gate where an extra unique event has been added. The tree was broken into smaller parts for easier loading into SAPHIRE. The tree, TEST3 under the Family named GATES, was constructed by combining the broken parts of the large tree with an AND gate. The tree, TEST3, contained 9,999 gates. A 10,000th gate is added in the build menu of SAPHIRE.

Acceptance Criteria

The cut sets for SAPHIRE and CAFTA should match and the maximum number of gates should be 9,999. The cut sets produced for the maximum gate analysis should contain two single cut sets representing the two events that are the same for each gate plus a cutset containing the product of the unique events added to each top gate. The number of term events in the product cut set should be equivalent the number of "smaller part" trees ANDed in TEST3 under the family GATES.

Results/Anomaly Description

The tests passed the acceptance criteria with the exception of a modified run of the maximum gate tree that had just the same two distinct events under each gate, including the top gate. When this tree was analyzed using the "Generate Cut Set" option, there were two second order cut sets produced that contained the product of one of the two events with an event titled "Undeveloped Gate." The solution should have been two first order cut sets, one for each of the two events. SAPHIRE collapsed this tree incorrectly. Checking the modified logic under the REPORT-SYSTEM-LOGIC menu showed that SAPHIRE had collapsed each branch ("smaller part") into temporary gates that had the same names and then combined the temporary gates with the same name in an AND gate. Combining the same gates in the AND gate caused SAPHIRE to produce an erroneous "Undeveloped Gate" in the final solution under the ANALYZE-Generate menu.

This discrepancy is categorized as critical because the results are incorrect. Even though it would not be common for a fault tree to be composed entirely of gates with the same events

connected by OR gates, especially not one this size, it cannot be ruled out entirely. The results would also be incorrect for a small tree of only two or three levels that consist of gates with the same events.

Input Test Data File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-2. SOFTWARE TEST RECORD — STR-22-000-0894

Date: September 4, 1994
Originator: James L. Jones, Jr.
Computer System: ALR 486/33-MHz BV, 17-MB RAM, 340-MB HD, XGA Display, DOS 6.21

Test Description

Cut set cut off value and Event Probability cutoff value. Do not discard cut set if the cut set failure probability is greater than the cutoff or failure probability for each event is greater than or equal to the event probability cutoff. To accomplish this Test, I used the Peach Bottom sequence A17 and generated cut sets for four different probability cutoff values (1E−2, 1E−9, 1E−11, 5E−12) for testing the cut set cutoff and 3.0E−3 event cutoff with a 1E−9 cut set cutoff for event cutoff test. This test was run substituting the quantified fault tree cut sets into the event tree top events and then run again substituting the fault tree logic into the top events.

Acceptance Criteria

The expected values would be consistent with the probability cutoff and event cutoff values in the reports.

Results/Anomaly Description

The cut set failure probability cutoff test passed the acceptance criteria. The event cutoff test failed the acceptance criteria. This failure occurred when comparing the results for the method quantifying the event tree using fault tree cut set substitution with the results obtained by quantifying the same event tree using fault tree logic substitution. Each method produced the identical number (13) and composition of cut sets when quantified under the cut set probability cut off test. However, when quantified using the event cutoff feature, the quantified fault tree substitution method produced 769 cut sets while the fault tree logic substitution method produced 14 cut sets. The first 13 cut sets for each method were identical; however, the fourteenth cut set for the quantified fault tree method did not match the fourteenth cut set for the fault tree logic method. This fourteenth cut set for the fault tree logic method matched the fifteenth cut set for the quantified fault tree method. Furthermore, the fourteenth cut set for the quantified fault tree method was a legitimate cut set. By this statement I mean that each event in the cut set had a value greater than or equal to 3E−3 and this cut set appeared in the list of cut sets generated by the cut set failure probability cutoff test for a cutoff value of 1E−11. Therefore, one or both of the results produced in the event cutoff test are erroneous.

This discrepancy is categorized as critical. It should be noted that this feature is not commonly used; however, the results are still incorrect.

Input Test Data File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-3. SOFTWARE TEST RECORD — 45-01A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the lognormal distribution for a single basic event. Six different cases were tested for the lognormal distribution. The six cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
Uncertainty value	10	10	1,000	1,000	1,000	10

Acceptance Criteria

The generated results must be equal to the expected values for the six distributions. The expected values for the six distributions are:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
5th percentile	3.75E−6	3.75E−8	1.48E−22	1.48E−23	1.48E−24	3.75E−2
50th percentile	3.75E−5	3.75E−7	1.48E−19	1.48E−20	1.48E−21	3.75E−1
95th percentile	3.75E−4	3.75E−6	1.48E−16	1.48E−17	1.48E−18	3.75E+0
Standard dev.	2.47E−4	2.47E−6	6.75E−12	6.75E−13	6.75E−14	2.47E+0

Results/Anomaly Description

Out of the six cases tested, one noncritical and one critical anomalies were noted. The noncritical anomaly was noted for test Cases C, D, and E, which were lognormal distributions with a mean value equal to or less than 1.0E−15. For these test cases, a comparison of the expected values and obtained values indicates that the code appears to treat very small sampled values as zeroes in determining the distribution. Also, the point estimate values are incorrect. This test represents a boundary test. The critical anomaly was noted for test Case F, which was a lognormal distribution with a mean value = 1 and an error factor = 10. Although the basic event was indicated

as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a lognormal distribution with the following input parameters:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean value	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
Uncertainty value	10	10	1,000	1,000	1,000	10
Calculation type	1	1	1	1	1	1
Distribution type	L	L	L	L	L	L
Initiating event	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-4. SOFTWARE TEST RECORD — 45-02A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the normal distribution for a single basic event. Two different normal distributions were tested with 15 cases being tested for each distribution. The two different normal distributions and 15 cases tested included:

	Distribution A (mean value = 0) (standard deviation = 1)		Distribution B (mean value = 0.5) (standard deviation = 1)	
	Seed value	Sample size	Seed value	Sample size
Case A	512	500	512	500
Case B	512	1,000	512	1,000
Case C	512	3,000	512	3,000
Case D	512	5,000	512	5,000
Case E	512	10,000	512	10,000
Case F	4,321	500	4,321	500
Case G	4,321	1,000	4,321	1,000
Case H	4,321	3,000	4,321	3,000
Case I	4,321	5,000	4,321	5,000
Case J	4,321	10,000	4,321	10,000
Case K	65,533	500	65,533	500
Case L	65,533	1,000	65,533	1,000
Case M	65,533	3,000	65,533	3,000
Case N	65,533	5,000	65,533	5,000
Case O	65,533	10,000	65,533	10,000

Acceptance Criteria

The generated results for the 15 cases for each of the two distributions tested must be equal to the expected values for the distributions. The expected values for the distributions are:

Parameter	Distribution A	Distribution B
Mean	0	0.5
5th percentile	−1.65	−1.15
50th percentile	0	0
95th percentile	1.65	2.15
Standard deviation	1	1

Results/Anomaly Description

Out of the two distributions tested, two critical anomalies were noted. The first anomaly noted was for Distribution A, which was a normal distribution with a mean value = 0 and a standard deviation = 1 (standard normal distribution). Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 0 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing normal distributions. A mean frequency value = 0 is acceptable for a normal distribution and uncertainty should not be ignored. The second anomaly was noted for a separate test that was run for Distribution B with the basic event not being indicated as an initiating event. For this separate test, the generated distribution was truncated at 1.0 but was not truncated at 0.0 (negative probability values were allowed).

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a normal distribution with the following input parameters:

Parameter	Distribution A	Distribution B
Mean value	0	0.5
Uncertainty value	1	1
Calculation type	1	1
Distribution type	N	N
Initiating event	Y	Y

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-5. SOFTWARE TEST RECORD — 45-03A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the beta distribution for a single basic event. Ten different cases were tested for the beta distribution. The cases tested included:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	0.5	1.0	−1.0	0.5	1E−3	1E−6	1E−6	0.5	0.5	10.0
Uncertainty value	0.5	1.0	0.5	0.0	0.2	2.0	8E+5	1.0	5.0	2.0

Acceptance Criteria

The generated results must be equal to the expected values for the 10 distributions. The expected values for the distributions are:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	0.5	1.0	n/a	n/a	0.001	1E−6	1E−6	0.5	0.5	n/a
Standard dev.	0.354	0.0	n/a	n/a	0.0289	5.8E−4	1.1E−6	0.289	0.151	n/a

Results/Anomaly Description

Out of the 10 cases tested, three critical anomalies were noted. The first anomaly was noted for test Cases E, F, and G, which were beta distributions with mean values equal to or less than 1E−3 and various uncertainty values. The results for these three cases are suspect due to the 0.0 results that are obtained. In fact, a standard deviation value = 0.0 was obtained for Case F when a standard deviation value = 5.77E−4 was expected. For these test cases, the code appears to treat very small sampled values as zeroes in determining the distribution. The second anomaly was noted for Case H, which was a beta distribution with a mean value = 0.5 and an uncertainty value = 1. When modifying the event data, a message at the bottom of the screen indicates that a b value = 1 for a beta distribution implies no uncertainty and that any uncertainty data will be ignored. This is an incorrect statement and the obtained results should be equivalent to a uniform distribution that has a mean value = 0.5 and a high value = 1. The third anomaly was noted for test Case J, which

was a beta distribution with a mean value = 10 and an uncertainty value = 2. These input parameters are not possible for a beta distribution. The generated results for test Case J are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a beta distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean value	0.5	1.0	−1.0	0.5	1E−3	1E−6	1E−6	0.5	0.5	10.0
Uncertainty value	0.5	1.0	0.5	0.0	0.2	2.0	8E+5	1.0	5.0	2.0
Calculation type	1	1	1	1	1	1	1	1	1	1
Distribution type	B	B	B	B	B	B	B	B	B	B
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-6. SOFTWARE TEST RECORD — 45-04A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the chi-square distribution for a single basic event. Ten different cases were tested for the chi-square distribution. The cases tested included:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	2	5	4	0.1	40,000	100	0	0	0	1
Uncertainty value	1	1	2	5	5	100	0	1.5	1	1

Acceptance Criteria

The generated results must be equal to the expected values for the nine distributions. The expected values for the distributions were based upon the following criteria (according to the *SAPHIRE Version 5.0 Technical Reference Manual*, page 71):

- Distribution A should be distributed as $2\chi^2(1)$
- Distribution B should be distributed as $5\chi^2(1)$
- Distribution C should be distributed as $2\chi^2(2)$
- Distribution D should be distributed as $(1/50)\chi^2(5)$
- Distribution E should be distributed as $8000\chi^2(5)$
- Distribution F should be distributed as $\chi^2(100)$
- Distribution G should be undefined
- Distribution H should be zero
- Distribution I should be zero
- Distribution J should be distributed as $\chi^2(1)$.

Results/Anomaly Description

Out of the 10 cases tested, one critical anomaly was noted. The anomaly was noted for Case J, which was a chi-square distribution with a mean value = 1 and an uncertainty value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for test Case J are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a chi-square distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean value	2	5	4	0.1	40,000	100	0	0	0	1
Uncertainty value	1	1	2	5	5	100	0	1.5	1	1
Calculation type	1	1	1	1	1	1	1	1	1	1
Distribution type	C	C	C	C	C	C	C	C	C	C
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-7. SOFTWARE TEST RECORD — 45-04B

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the chi-square distribution for a single basic event. Three different chi-square distributions were tested with five cases being tested for each distribution. The three different chi-square distributions and five cases tested include:

	Distribution A (mean value = 1) (uncertainty value = 1)	Distribution B (mean value = 2) (uncertainty value = 2)	Distribution C (mean value = 5) (uncertainty value = 5)
	Sample size	Sample size	Sample size
Case A	500	500	500
Case B	1,000	1,000	1,000
Case C	3,000	3,000	3,000
Case D	5,000	5,000	5,000
Case E	10,000	10,000	10,000

Acceptance Criteria

The generated results must be equal to the expected values for the distributions. The expected values for the distributions were based upon the following criteria (according to the *SAPHIRE Version 5.0 Technical Reference Manual*, page 71):

- Distribution A should be distributed as $\chi^2(1)$
- Distribution B should be distributed as $\chi^2(2)$
- Distribution C should be distributed as $\chi^2(5)$.

Results/Anomaly Description

Out of the three distributions tested, one critical anomaly was noted. The one anomaly noted was for Distribution A (all five cases), which was a chi-square distribution with a mean value = 1 and an uncertainty value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any

uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for the five cases tested for Distribution A are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a chi-square distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C
Mean value	1	2	5
Uncertainty value	1	2	5
Calculation type	1	1	1
Distribution type	C	C	C
Initiating event	Y	Y	Y
Seed number	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-8. SOFTWARE TEST RECORD — 45-05A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the exponential distribution for a single basic event. Eight different cases were tested for the exponential distribution. The eight cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F	Case G	Case H
Mean	−1.0	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16

Acceptance Criteria

The generated results must be equal to the expected values for the eight distributions. The expected values for the eight distributions are:

Parameter	A	B	C	D	E	F	G	H
Mean	−1.0	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16
5th percentile	n/a	0.0	5.13E−3	5.13E−2	2.565	2.565E+2	1.0E−5	7.4E−18
50th percentile	n/a	0.0	6.93E−2	6.93E−1	34.65	3.465E+3	1.39E−4	1.0E−16
95th percentile	n/a	0.0	2.996E−1	2.996	149.8	1.498E+4	5.99E−4	4.3E−16
Standard dev.	n/a	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16
Skewness	n/a	0.0	2.0	2.0	2.0	2.0	2.0	2.0
Kurtosis	n/a	0.0	9.0	9.0	9.0	9.0	9.0	9.0

Results/Anomaly Description

Out of the eight cases tested, one critical and one noncritical anomalies were noted. The critical anomaly was noted for test Case D, which was an exponential distribution with a mean value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen

indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The noncritical anomaly was noted for test Case H, which was an exponential distribution with a mean value = 1.443E−16. For this test case, a comparison of the expected values and obtained values indicates that the code appears to treat very small sampled values as zeroes in determining the distribution. This test represents a boundary test.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had an exponential distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H
Mean value	−1.0	0.0	0.1	1.0	50.0	5.0E+3	2.0E−4	1.443E−16
Uncertainty value	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Calculation type	1	1	1	1	1	1	1	1
Distribution type	E	E	E	E	E	E	E	E
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-9. SOFTWARE TEST RECORD — 45-06A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the uniform distribution for a single basic event. Four different cases were tested for the uniform distribution. The four cases tested included:

Parameter	Case A	Case B	Case C	Case D
Mean	0.5	50,000	0.0	1.0
Low value	0.0	0.0	−1.0	0.0
High value	1.0	100,000	1.0	2.0

Acceptance Criteria

The generated results must be equal to the expected values for the four distributions. The expected values for the four distributions are:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Mean	0.5	50,000	0.0	1.0
5th percentile	0.05	5,000	−0.1	0.1
50th percentile	0.5	50,000	0.0	1.0
95th percentile	0.95	95,000	0.9	1.9
Standard deviation	0.2887	28,868	0.5774	0.5774
Skewness	0.0	0.0	0.0	0.0
Kurtosis	1.8	1.8	1.8	1.8

Results/Anomaly Description

From the four cases tested, two critical anomalies were noted. The first anomaly was noted for test Case C, which was a uniform distribution with a mean value = 0 and a low value = −1. The code does not allow the distribution to have negative values. The second anomaly was noted for test Case D, which was a uniform distribution with a mean value = 1 and a low value = 0. Although the

basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a uniform distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Mean value	0.5	50,000	0.0	1.0
Upper end value	1.0	100,000	1.0	2.0
Calculation type	1	1	1	1
Distribution type	U	U	U	U
Initiating event	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-10. SOFTWARE TEST RECORD — 45-07A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the histogram distribution for a single basic event. Four different cases were tested for the histogram distribution. The four cases tested included:

Parameter	Case A	Case B	Case C	Case D
Bin/probability (percent/area/height)	1/0.1 (10) 2/0.2 (10) 3/0.3 (10) 4/0.4 (10) 5/0.5 (10) 6/0.6 (10) 7/0.7 (10) 8/0.8 (10) 9/0.9 (10) 10/1.0 (10)	1/0.5 (100)	0/1.0E−3 1/1.0E−2 (0.5) 2/1.0E−1 (0.5)	0/1.0E−3 1/1.0E−2 (55.55) 2/1.0E−1 (5.555)
Histogram type	Percentage	Percentage	Area	Range

Acceptance Criteria

The generated results must be equal to the expected values for the four distributions. The expected values for the distributions were based on the following criteria:

- Distribution A should resemble a uniform distribution from 0 to 1
- Distribution B should be a point estimate with a value of 0.5
- Distribution C should have a mean value approximately = 0.0248, a standard deviation value approximately = 0.0202, and be skewed to the right
- Distribution D should resemble Distribution C.

Results/Anomaly Description

Out of the four cases tested, one noncritical anomaly and one critical anomaly were noted. The noncritical anomaly was noted when printing a histogram detail report. For the area

(HIST3—Distribution C) and range (HIST4—Distribution D) histograms, the values listed are under the wrong headings. The critical anomaly was noted for all the cases tested. The critical anomaly is related to the mean value that is input when modifying the basic event data. The code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram being evaluated should be determined by the code and used; otherwise, incorrect point estimates will be calculated. Also, if the mean value is left blank or if a mean value = 1 or 0 is entered, the message at the bottom of the screen indicates that a frequency value = 1, 0, or blank implies no uncertainty and any uncertainty data will be ignored. The histogram that has been entered is ignored by the code.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a histogram distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Bin/probability (percent/area/height)	1/0.1 (10) 2/0.2 (10) 3/0.3 (10) 4/0.4 (10) 5/0.5 (10) 6/0.6 (10) 7/0.7 (10) 8/0.8 (10) 9/0.9 (10) 10/1.0 (10)	1/0.5 (100)	0/1.0E−3 1/1.0E−2 (0.5) 2/1.0E−1 (0.5)	0/1.0E−3 1/1.0E−2 (55.55) 2/1.0E−1 (5.555)
Histogram type	Percentage	Percentage	Area	Range
Calculation type	1	1	1	1
Distribution type	H	H	H	H
Initiating event	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321

Results file name(s) and/or output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-11. SOFTWARE TEST RECORD — 45-08A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the gamma distribution for a single basic event. Six different cases were tested for the gamma distribution. The six cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Uncertainty value	5	20	1E+4	100	0.5	1

Acceptance Criteria

The generated results must be equal to the expected values for the six distributions. The expected values for the six distributions are:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Standard deviation	0.2236	22.36	1E−6	1E−3	1.4142	2
Skewness	0.894	0.447	0.02	0.2	2.83	2
Kurtosis	4.2	3.3	3.0006	3.06	15	9

Results/Anomaly Description

Out of the six cases tested, one critical anomaly was noted. The anomaly was noted for Case E, which was a gamma distribution with a mean value = 1 and an uncertainty value = 0.5. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for test Case E are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a gamma distribution with the following input parameters:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Uncertainty value	5	20	1E+4	100	0.5	1
Calculation type	1	1	1	1	1	1
Distribution type	G	G	G	G	G	G
Initiating event	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-12. SOFTWARE TEST RECORD — 45-09A

Date: October 5, 1994
Originator: Curtis L. Smith
Computer System: ISC 66-MHz 486, 32-MB RAM, 512-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for the Maximum Entropy (ME) distribution for a single basic event using various distribution parameters. The various parameters for the ME distribution include:

	Mean	Lower End	Upper End
Case A	5.000E−01	0.000E+00	1.000E+00
Case B	2.500E−01	0.000E+00	1.000E+00
Case C	7.500E−01	0.000E+00	1.000E+00
Case D	1.000E−15	0.000E+00	1.000E+00
Case E	9.999E−01	0.000E+00	1.000E+00
Case F	4.999E−01	0.000E+00	1.000E+00
Case G	5.001E−01	0.000E+00	1.000E+00

Acceptance Criteria

The resulting uncertainty distribution should vary depending on the parameters specified for the distribution. If the mean value is the midpoint between the lower and upper end, the distribution should be uniform. If the mean is closer to the lower end, the distribution will be a truncated exponential distribution. If the mean is closer to the upper end, the distribution will be a truncated "reverse" exponential distribution (i.e., the probability density function increases from left to right). The uncertainty results were checked for each case above to determine if the case appeared to match the appropriate distribution. These results are attached.

Results/Anomaly Description

Out of the seven cases tested, two critical and one noncritical anomalies appeared.

Case A Critical. The results for this case returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.5. The standard deviation turned out to be 0 instead of the correct value of 0.2887.

Case D Noncritical—boundary test. This case had a very low mean value (1E−15). The uncertainty returned a point estimate value of 9.992E−016, which is incorrect (the correct mincut should be 1E−15). Also, some of the sample values were zero. This implies that we have reached the limits of the calculational accuracy. Also, this one case took 22.19 seconds while the other six cases took between 10.43 and 10.49 seconds (why the elapsed time for this one case is over twice as long as the other cases is unknown).

Case E Critical. This case had a mean value close to the upper end point (mean = 0.9999 and upper end = 1.0). The uncertainty results for this case returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.9999. The standard deviation turned out to be 0 while the skewness and kurtosis were 1.0E+38.

Input Test Data File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-13. SOFTWARE TEST RECORD — 46-01A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the lognormal distribution for a single basic event. Six different cases were tested for the lognormal distribution. The six cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
Uncertainty value	10	10	1,000	1,000	1,000	10

Acceptance Criteria

The generated results must be equal to the expected values for the six distributions. The expected values for the six distributions are:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
5th percentile	3.75E−6	3.75E−8	1.48E−22	1.48E−23	1.48E−24	3.75E−2
50th percentile	3.75E−5	3.75E−7	1.48E−19	1.48E−20	1.48E−21	3.75E−1
95th percentile	3.75E−4	3.75E−6	1.48E−16	1.48E−17	1.48E−18	3.75E+0
Standard dev.	2.47E−4	2.47E−6	6.75E−12	6.75E−13	6.75E−14	2.47E+0

Results/Anomaly Description

Out of the six cases tested, one noncritical and one critical anomalies were noted. The noncritical anomaly was noted for test Cases C, D, and E, which were lognormal distributions with a mean value equal to or less than 1.0E−15. For these test cases, a comparison of the expected values and obtained values indicates that the code appears to treat very small sampled values as zeroes in determining the distribution. Also, the point estimate values are incorrect. This test represents a boundary test. The critical anomaly was noted for test Case F, which was a lognormal distribution with a mean value = 1 and an error factor = 10. Although the basic event was indicated

as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a lognormal distribution with the following input parameters:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean value	1E−4	1E−6	1E−15	1E−16	1E−17	1.0
Uncertainty value	10	10	1,000	1,000	1,000	10
Calculation type	1	1	1	1	1	1
Distribution type	L	L	L	L	L	L
Initiating event	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-14. SOFTWARE TEST RECORD — 46-03A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the beta distribution for a single basic event. Ten different cases were tested for the beta distribution. The cases tested included:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	0.5	1.0	−1.0	0.5	1E−3	1E−6	1E−6	0.5	0.5	10.0
Uncertainty value	0.5	1.0	0.5	0.0	0.2	2.0	8E+5	1.0	5.0	2.0

Acceptance Criteria

The generated results must be equal to the expected values for the 10 distributions. The expected values for the distributions are:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	0.5	1.0	n/a	n/a	0.001	1E−6	1E−6	0.5	0.5	n/a
Standard dev.	0.354	0.0	n/a	n/a	0.0289	5.8E−4	1.1E−6	0.289	0.151	n/a

Results/Anomaly Description

Out of the 10 cases tested, three critical anomalies were noted. The first anomaly was noted for test Cases F and G, which were beta distributions with mean values = 1E−6 and uncertainty values = 2.0 and 8E+5, respectively. The generated results for these two cases are incorrect. The second anomaly was noted for Case H, which was a beta distribution with a mean value = 0.5 and an uncertainty value = 1. When modifying the event data, a message at the bottom of the screen indicates that a b value = 1 for a beta distribution implies no uncertainty and that any uncertainty data will be ignored. This is an incorrect statement and the obtained results should be equivalent to a uniform distribution that has a mean value = 0.5 and a high value = 1. The third anomaly was noted for test Case J, which was a beta distribution with a mean value = 10 and an uncertainty value = 2. These input parameters are not possible for a beta distribution. Run-time errors resulting in the need to recover the database occurred when the uncertainty analysis was performed for this case.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a beta distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean value	0.5	1.0	−1.0	0.5	1E−3	1E−6	1E−6	0.5	0.5	10.0
Uncertainty value	0.5	1.0	0.5	0.0	0.2	2.0	8E+5	1.0	5.0	2.0
Calculation type	1	1	1	1	1	1	1	1	1	1
Distribution type	B	B	B	B	B	B	B	B	B	B
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-15. SOFTWARE TEST RECORD — 46-04A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the chi-square distribution for a single basic event. Ten different cases were tested for the chi-square distribution. The cases tested included:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean	2	5	4	0.1	40,000	100	0	0	0	1
Uncertainty value	1	1	2	5	5	100	0	1.5	1	1

Acceptance Criteria

The generated results must be equal to the expected values for the nine distributions. The expected values for the distributions were based on the following criteria (according to the *SAPHIRE Version 5.0 Technical Reference Manual*, page 71):

- Distribution A should be distributed as $2\chi^2(1)$
- Distribution B should be distributed as $5\chi^2(1)$
- Distribution C should be distributed as $2\chi^2(2)$
- Distribution D should be distributed as $(1/50)\chi^2(5)$
- Distribution E should be distributed as $8000\chi^2(5)$
- Distribution F should be distributed as $\chi^2(100)$
- Distribution G should be undefined
- Distribution H should be zero
- Distribution I should be zero
- Distribution J should be distributed as $\chi^2(1)$.

Results/Anomaly Description

Out of the 10 cases tested, one critical anomaly was noted. The anomaly was noted for Case J, which was a chi-square distribution with a mean value = 1 and an uncertainty value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for test Case J are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a chi-square distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H	I	J
Mean value	2	5	4	0.1	40,000	100	0	0	0	1
Uncertainty value	1	1	2	5	5	100	0	1.5	1	1
Calculation type	1	1	1	1	1	1	1	1	1	1
Distribution type	C	C	C	C	C	C	C	C	C	C
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-16. SOFTWARE TEST RECORD — 46-04B

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the chi-square distribution for a single basic event. Three different chi-square distributions were tested with five cases being tested for each distribution. The three different chi-square distributions and five cases tested include:

	Distribution A (mean value = 1) (uncertainty value = 1)	Distribution B (mean value = 2) (uncertainty value = 2)	Distribution C (mean value = 5) (uncertainty value = 5)
	Sample size	Sample size	Sample size
Case A	500	500	500
Case B	1,000	1,000	1,000
Case C	3,000	3,000	3,000
Case D	5,000	5,000	5,000
Case E	10,000	10,000	10,000

Acceptance Criteria

The generated results must be equal to the expected values for the distributions. The expected values for the distributions were based upon the following criteria (according to the *SAPHIRE Version 5.0 Technical Reference Manual*, page 71):

- Distribution A should be distributed as $\chi^2(1)$
- Distribution B should be distributed as $\chi^2(2)$
- Distribution C should be distributed as $\chi^2(5)$.

Results/Anomaly Description

Out of the three distributions tested, one critical anomaly was noted. The one anomaly noted was for Distribution A (all five cases), which was a chi-square distribution with a mean value = 1 and an uncertainty value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any

uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for the five cases tested for Distribution A are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a chi-square distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C
Mean value	1	2	5
Uncertainty value	1	2	5
Calculation type	1	1	1
Distribution type	C	C	C
Initiating event	Y	Y	Y
Seed number	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-17. SOFTWARE TEST RECORD — 46-05A

Date: January 19, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the exponential distribution for a single basic event. Eight different cases were tested for the exponential distribution. The eight cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F	Case G	Case H
Mean	−1.0	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16

Acceptance Criteria

The generated results must be equal to the expected values for the eight distributions. The expected values for the eight distributions are:

Parameter	A	B	C	D	E	F	G	H
Mean	−1.0	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16
5th percentile	n/a	0.0	5.13E−3	5.13E−2	2.565	2.565E+2	1.0E−5	7.4E−18
50th percentile	n/a	0.0	6.93E−2	6.93E−1	34.65	3.465E+3	1.39E−4	1.0E−16
95th percentile	n/a	0.0	2.996E−1	2.996	149.8	1.498E+4	5.99E−4	4.3E−16
Standard dev.	n/a	0.0	0.1	1.0	50	5.0E+3	2.0E−4	1.443E−16
Skewness	n/a	0.0	2.0	2.0	2.0	2.0	2.0	2.0
Kurtosis	n/a	0.0	9.0	9.0	9.0	9.0	9.0	9.0

Results/Anomaly Description

Out of the eight cases tested, one critical and one noncritical anomaly were noted. The critical anomaly was noted for test Case D, which was an exponential distribution with a mean value = 1. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The noncritical anomaly was noted for test Case H, which was an exponential distribution with a mean value = 1.443E−16. For this test case, a comparison of the expected values and obtained values indicates that the code appears to treat very small sampled values as zeroes in determining the distribution. Also, the point estimate value is incorrect. This test represents a boundary test.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had an exponential distribution with the following input parameters:

Parameter	A	B	C	D	E	F	G	H
Mean value	−1.0	0.0	0.1	1.0	50.0	5.0E+3	2.0E−4	1.443E−16
Uncertainty value	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Calculation type	1	1	1	1	1	1	1	1
Distribution type	E	E	E	E	E	E	E	E
Initiating event	Y	Y	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-18. SOFTWARE TEST RECORD — 46-06A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the uniform distribution for a single basic event. Four different cases were tested for the uniform distribution. The four cases tested included:

Parameter	Case A	Case B	Case C	Case D
Mean	0.5	50,000	0.0	1.0
Low value	0.0	0.0	−1.0	0.0
High value	1.0	100,000	1.0	2.0

Acceptance Criteria

The generated results must be equal to the expected values for the four distributions. The expected values for the four distributions are:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Mean	0.5	50,000	0.0	1.0
5th percentile	0.05	5,000	−0.1	0.1
50th percentile	0.5	50,000	0.0	1.0
95th percentile	0.95	95,000	0.9	1.9
Standard deviation	0.2887	28,868	0.5774	0.5774
Skewness	0.0	0.0	0.0	0.0
Kurtosis	1.8	1.8	1.8	1.8

Results/Anomaly Description

From the four cases tested, two critical anomalies were noted. The first anomaly was noted for test Case C, which was a uniform distribution with a mean value = 0 and a low value = −1. The code does not allow the distribution to have negative values. The second anomaly was noted for test Case D, which was a uniform distribution with a mean value = 1 and a low value = 0. Although the

basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a uniform distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Mean value	0.5	50,000	0.0	1.0
Upper end value	1.0	100,000	1.0	2.0
Calculation type	1	1	1	1
Distribution type	U	U	U	U
Initiating event	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-19. SOFTWARE TEST RECORD — 46-07A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the histogram distribution for a single basic event. Four different cases were tested for the histogram distribution. The four cases tested included:

Parameter	Case A	Case B	Case C	Case D
Bin/probability (percent/area/height)	1/0.1 (10) 2/0.2 (10) 3/0.3 (10) 4/0.4 (10) 5/0.5 (10) 6/0.6 (10) 7/0.7 (10) 8/0.8 (10) 9/0.9 (10) 10/1.0 (10)	1/0.5 (100)	0/1.0E−3 1/1.0E−2 (0.5) 2/1.0E−1 (0.5)	0/1.0E−3 1/1.0E−2 (55.55) 2/1.0E−1 (5.555)
Histogram type	Percentage	Percentage	Area	Range

Acceptance Criteria

The generated results must be equal to the expected values for the four distributions. The expected values for the distributions were based on the following criteria:

- Distribution A should resemble a uniform distribution from 0 to 1
- Distribution B should be a point estimate with a value of 0.5
- Distribution C should have a mean value approximately = 0.0248, a standard deviation value approximately = 0.0202, and be skewed to the right
- Distribution D should resemble Distribution C.

Results/Anomaly Description

Out of the four cases tested, one noncritical anomaly and one critical anomaly were noted. The noncritical anomaly was noted when printing a histogram detail report. For the area

(HIST3—Distribution C) and range (HIST4—Distribution D) histograms, the values listed are under the wrong headings. The critical anomaly was noted for all the cases tested. The critical anomaly is related to the mean value that is input when modifying the basic event data. The code will allow any mean value to be entered and uses this value for the point estimate. The mean value from the histogram being evaluated should be determined by the code and used; otherwise, incorrect point estimates will be calculated. Also, if the mean value is left blank or if a mean value = 1 or 0 is entered, the message at the bottom of the screen indicates that a frequency value = 1, 0, or blank implies no uncertainty and any uncertainty data will be ignored. The histogram that has been entered is ignored by the code.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a histogram distribution with the following input parameters:

Parameter	Distribution A	Distribution B	Distribution C	Distribution D
Bin/probability (percent/area/height)	1/0.1 (10) 2/0.2 (10) 3/0.3 (10) 4/0.4 (10) 5/0.5 (10) 6/0.6 (10) 7/0.7 (10) 8/0.8 (10) 9/0.9 (10) 10/1.0 (10)	1/0.5 (100)	0/1.0E−3 1/1.0E−2 (0.5) 2/1.0E−1 (0.5)	0/1.0E−3 1/1.0E−2 (55.55) 2/1.0E−1 (5.555)
Histogram type	Percentage	Percentage	Area	Range
Calculation type	1	1	1	1
Distribution type	H	H	H	H
Initiating event	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-20. SOFTWARE TEST RECORD — 46-08A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the gamma distribution for a single basic event. Six different cases were tested for the gamma distribution. The six cases tested included:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Uncertainty value	5	20	1E+4	100	0.5	1

Acceptance Criteria

The generated results must be equal to the expected values for the six distributions. The expected values for the six distributions are:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Standard deviation	0.2236	22.36	1E−6	1E−3	1.4142	2
Skewness	0.894	0.447	0.02	0.2	2.83	2
Kurtosis	4.2	3.3	3.0006	3.06	15	9

Results/Anomaly Description

Out of the six cases tested, one noncritical and one critical anomaly were noted. The noncritical anomaly was noted for Case C, which was a gamma distribution with a mean value = 1E−4 and an uncertainty value = 1E+4. For this test case, the message "A too large, ITMAX too small in GSER (or GCF)" appeared at the bottom of the screen during the uncertainty calculation. Also, this one case took 30.26 seconds while the other five cases took between 8.73 and 13.95 seconds (why the elapsed time for this one case took over twice as long as the other cases is unknown). The critical anomaly was noted for Case E, which was a gamma distribution with a mean value = 1 and an uncertainty value = 0.5. Although the basic event was indicated as an initiating event, the message at the bottom of the screen indicates that a frequency value = 1 implies no uncertainty and any uncertainty data will be ignored. This is an incorrect statement when developing frequency

distributions. A frequency value = 1 is acceptable and uncertainty should not be ignored. The generated results for test Case E are incorrect.

Input Test Data File Name(s) and/or Output

One basic event was input to an OR gate. The basic event had a gamma distribution with the following input parameters:

Parameter	Case A	Case B	Case C	Case D	Case E	Case F
Mean	0.5	100	1E−4	1E−2	1	2
Uncertainty value	5	20	1E+4	100	0.5	1
Calculation type	1	1	1	1	1	1
Distribution type	G	G	G	G	G	G
Initiating event	Y	Y	Y	Y	Y	Y
Sample size	5,000	5,000	5,000	5,000	5,000	5,000
Seed number	4,321	4,321	4,321	4,321	4,321	4,321

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-21. SOFTWARE TEST RECORD — 46-09A

Date: January 18, 1995
Originator: Curtis L. Smith
Computer System: ISC 66-MHz 486, 32-MB RAM, 512-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for the Maximum Entropy (ME) distribution for a single basic event using various distribution parameters. The various parameters for the ME distribution are

	Mean	Lower End	Upper End
Case A	5.000E−01	0.000E+00	1.000E+00
Case B	2.500E−01	0.000E+00	1.000E+00
Case C	7.500E−01	0.000E+00	1.000E+00
Case D	1.000E−15	0.000E+00	1.000E+00
Case E	9.999E−01	0.000E+00	1.000E+00
Case F	4.999E−01	0.000E+00	1.000E+00
Case G	5.001E−01	0.000E+00	1.000E+00

Acceptance Criteria

The resulting uncertainty distribution should vary depending on the parameters specified for the distribution. If the mean value is the midpoint between the lower and upper end, the distribution should be uniform. If the mean is closer to the lower end, the distribution will be a truncated exponential distribution. If the mean is closer to the upper end, the distribution will be a truncated "reverse" exponential distribution (i.e., the probability density function increases from left to right). The uncertainty results were checked for each case above to determine if the case appeared to match the appropriate distribution. These results are attached.

Results/Anomaly Description

Out of the seven cases tested, two critical and one noncritical anomalies appeared.

Case A Critical. The results for this case returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.5. The standard deviation turned out to be 0 instead of the correct value 0.2887.

Case D Noncritical—boundary test. This case had a very low mean value (1E−15). The uncertainty returned a point estimate value of 9.992E−016 which is incorrect (the correct mincut should be 1E−15). Also, some of the sample values were zero. This implies that we have reached the limits of the calculational accuracy.

Case E Critical. This case had a mean value close to the upper end point (mean = 0.9999 and upper end = 1.0). The uncertainty results for this case returned values of 1.0 for each sample. Consequently, the mean turned out to be 1.0 instead of the correct value of 0.9999. The standard deviation turned out to be 0 while the skewness and kurtosis were 1.0E+38.

Input Test Data File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-22. SOFTWARE TEST RECORD — 47-04A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Monte Carlo sampling for multiple event sampling on all sequences as a group in a family. The Beaver Valley Unit 2 IPE family from the PRA Data Loading Project (FIN A6883) was the family tested.

Acceptance Criteria

The generated results must be equal to the results from the Beaver Valley Unit 2 IPE. It is expected that the generated core damage frequency (CDF) results will be slightly lower than the IPE CDF results because 30 flood-related dominant accident (core damage) sequences that were identified and included in the IPE were not included in the SAPHIRE database. A total of 1,667 sequences were included in the SAPHIRE database for this analysis. The expected CDF values from the IPE are:

Parameter	Frequency
Mean	1.9E−04
95th Percentile	3.3E−04
50th Percentile	1.6E−04
5th Percentile	9.4E−05

Results/Anomaly Description

The generated results appear to agree with the IPE results, considering that the generated results do not contain the 30 flood-related sequences included in the IPE results. Although the code passed the test for family sequence uncertainty analysis, one noncritical anomaly was noted. The anomaly was noted when marking the sequences for the analysis. If all the sequences were marked one at a time (i.e., with the F2 key) or in a range (i.e., with the F4 key) top to bottom, the analysis option only allowed single or group sequence analysis. If all the sequences were marked at once (i.e., with the F3 key), the analysis option allowed single, group, or family sequence analysis.

Input Test Data File Name(s) and/or Output

The BV2-4 database was evaluated with a sample size = 3,000 and a seed value = 4,321.

Results File Names(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-23. SOFTWARE TEST RECORD — 48-04A

Date: January 18, 1995
Originator: Michael B. Calley
Computer System: Gateway 2000 33-MHz 486, 16-MB RAM, 340-MB HD, SVGA, DOS 6.2

Test Description

Performed Latin Hypercube sampling for multiple event sampling on all sequences as a group in a family. The Beaver Valley Unit 2 IPE family from the PRA Data Loading Project (FIN A6883) was the family tested.

Acceptance Criteria

The generated results must be equal to the results from the Beaver Valley Unit 2 IPE. It is expected that the generated core damage frequency (CDF) results will be slightly lower than the IPE CDF results because 30 flood-related dominant accident (core damage) sequences that were identified and included in the IPE were not included in the SAPHIRE database. A total of 1,667 sequences were included in the SAPHIRE database for this analysis. The expected CDF values from the IPE are:

Parameter	Frequency
Mean	1.9E−04
95th Percentile	3.3E−04
50th Percentile	1.6E−04
5th Percentile	9.4E−05

Results/Anomaly Description

The generated results appear to agree with the IPE results, considering that the generated results do not contain the 30 flood-related sequences included in the IPE results. Although the code passed the test for family sequence uncertainty analysis, one noncritical anomaly was noted. The anomaly was noted when marking the sequences for the analysis. If all the sequences were marked one at a time (i.e., with the F2 key) or in a range (i.e., with the F4 key) top to bottom, the analysis option only allowed single or group sequence analysis. If all the sequences were marked at once (i.e., with the F3 key), the analysis option allowed single, group, or family sequence analysis.

Input Test Data File Name(s) and/or Output

The BV2-4 database was evaluated with a sample size = 3,000 and a seed value = 4,321.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

B-24. SOFTWARE TEST RECORD — STR-90-000-0994

Date: September 23, 1994
Originator: James L. Jones, Jr.
Computer System: ALR 486/33-MHz BV, 17-MB RAM, 340-MB HD, XGA Display, DOS 6.21

Test Description

The purpose of this test is to determine the validity of the database conversion routine for SAPHIRE 5.0. This test will use the LASALLE (LASALL-4) database loaded into SAPHIRE 4.0. One particular sequence and its events and cut sets, T100-LOSP, will be the object of examination. First, reports listing cut sets and frequencies and events ranked in order of importance are extracted from the SAPHIRE 4.0 database. Next the entire contents of the LASALL-4 directory from SAPHIRE 4.0 is moved into directory TEST90 under the SAF50 (SAPHIRE 5.0) directory. After starting SAPHIRE 5.0, the Select Family menu is accessed and the LASALL-4 family under directory TEST90 is selected. The response from SAPHIRE 5.0 should be:

> "Some of the data base files were created with a different version of the software. All conflicting files will be rebuilt and reorganized to match the current version. Data for this family cannot be accessed or modified until the files are rebuilt.
>
> Do you wish to continue |N| (Y/N)?"

The "Y" option is selected and the enter key is depressed. SAPHIRE 5.0 should then convert the 4.0 database format to the 5.0 database format. Next, reports listing cut sets and frequencies and events ranked in order of importance are extracted form the SAPHIRE 5.0 database. The results from the SAPHIRE 4.0 T100-LOSP are then compared with the results from the SAPHIRE 5.0 T100-LOSP sequence.

Acceptance Criteria

The results from the reports from the SAPHIRE 4.0 database for the T100-LOSP sequence should identically match the results from the SAPHIRE 5.0 database converted from the SAPHIRE 4.0 database for the T100-LOSP sequence.

Results/Anomaly Description

The test failed the acceptance criteria. The following is a selected list of counter examples in the comparison between SAPHIRE 4.0 and SAPHIRE 5.0 results for the T100-LOSP sequence:

SAPHIRE 4.0 Results	SAPHIRE 5.0 Results
No. of Cut Sets = 1128 CDF = 1.969E−005	No. of Cut Sets = 993 CDF = 1.825E−005
IMPORTANCE RANKING RESULTS:	
Occurrences of RA-8-1H = 614 Occurrences of RCICRMCOOL-FLAG = 927	Occurrences of RA-8-1H = 569 Occurrences of RCICRMCOOL-FLAG = 866

This discrepancy is categorized as critical. Even though the database can be extracted from SAPHIRE 4.0 as "flat files" and unloaded into SAPHIRE 5.0 (with the correct results being obtained), the data base conversion routine is the more common method used by most users.

Input Test Data File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Results File Name(s) and/or Output

Input test data and results are kept on file as part of the V&V project. Input test data and results can be requested from any of the authors.

Appendix C

Nonvital Features
Verification and Validation Anomaly Reports

Appendix C

Nonvital Features Verification and Validation Anomaly Reports

C-1 INTRODUCTION

There were three discrepancies identified when the SAPHIRE 5.0 nonvital features were tested. Anomaly reports were generated for each discrepancy and are provided below. In addition, the applicable procedures are also presented. The complete nonvital test design specification and test procedure documents for SAPHIRE 5.0 are kept on file and can be requested from any of the authors.

The anomaly report number is of the form:

AR-VFN-MMYY-XXX

where:

AR is the acronym for anomaly report
VFN is the two digit nonvital feature procedure number
MM is the two-digit code for the month of the test
YY is the two-digit code for the year of the test
XXX is the three-digit sequential number assigned by the tester.

The test record number is of the form:

IVVPRO-PN-SN-STN

where:

IVVPRO stands for integrated V&V procedure
PN is the two-digit code for the procedure number
SN is the two-digit code for the section number
STN is the two-digit code for the step number.

ANOMALY REPORT—AR-02-0194-001

Date: 01/12/94
Originator: Dawnie Judd
Test Record No.: IVVPRO-02-04-034

Source/Location

Procedure 2, Section 4, Step 34.

Description

Changing fonts. The "FONT" would not change in "EDIT TEXT." It was necessary to go back to "TEXT" and change the "FONT" in that menu.

Cause

Unknown.

Impact

Noncritical.

ANOMALY REPORT—AR-03-0194-002

Date: 1/13/94
Originator: Dawnie Judd
Test Record No.: IVVPR0-03-04-07 and 08

Source/Location

Procedure 3, Section 4, Steps 7 and 8.

Description

Using the "←Xfer" option—did not transfer back to the file "CONT."

Cause

Unknown.

Impact

Noncritical.

ANOMALY REPORT—AR-03-0194-003

Date: 01/13/94
Originator: Dawnie Judd
Test Record No.: IVVPRO-03-04-22, 23, 24

Source/Location

Procedure 3, Section 4, Steps 22, 23 and 24.

Description

In "Attributes" when entering the new text size, the "name" size did not change.

Cause

Unknown.

Impact

Noncritical.

C-2. IRRAS NONVITAL EVENT TREE FEATURES: PROCEDURE 2—SECTION 4

1. Place the cursor on the "Create Event Tree" option and press **<enter>** to return to the *Primary Event Tree Graphics* menu. Place the cursor on the "0.50" option and press the **left mouse button**. The message **"Enter new text size"** will be displayed in the bottom left corner of the screen. Type 1.5 and press **<enter>**. Verify the change on the *Primary Event Tree Graphics* menu.

2. Place the cursor on the "FILE" option and press the **left mouse button** to access the *FILE* menu. Place the cursor on the "Load" option and press the **left mouse button**. The message **"Enter file name>"** will be displayed in the bottom left corner of the screen. Type **LOCA** and press **<enter>**. Verify that the event tree appears on the screen.

3. Place the cursor on the "TEXT" option and press the **left mouse button** to access the *TEXT* menu.

4. Place the cursor on the "Write" option and press the **left mouse button**. The message **"Pick the text placement location"** will be displayed in the bottom left corner of the screen.

5. Place the cursor under the event tree and press the **left mouse button** to access the text window. Type **Figure F-3. Large LOCA event tree** and press the **<ESC>** key . Verify that the text appears on the screen. Press the **right mouse button** to cancel the "Write" option.

6. Return to the *Primary Event Tree Graphics* menu. Place the cursor on the "FILE" option and press the **left mouse button** to access the *FILE* menu. Place the cursor on the "Save" option and press the **left mouse button**. The message **"Enter file name or CR for LOCA.ETG"** will be displayed in the bottom left corner of the screen.

7. Press **<enter>** and the message **"File already exists! Replace file? <Y/N>"** will be displayed in the bottom left corner of the screen. Type **Y** to save the file. Place the cursor on the *FILE* menu and press the **right mouse button** to cancel this menu.

8. Place the cursor on the "FONT" option and press the **left mouse button** to access the *FONT* menu. Place the cursor on one of the font options and press the **left mouse button**.

9. Return to the *Text* menu. Place the cursor on the "Write" option and press the **left mouse button**. The message **"Pick the text placement location"** will be displayed in the bottom left corner of the screen.

10. Place the cursor at a location under the event tree. Press the **left mouse button** to access the text window. Type text of your choice and press the **<ESC>** key. Verify that the text changed to the desired font on the screen. Press the **right mouse button** to terminate the "Write" option.

11. Repeat Steps 8 through 10 for several fonts. Verify that the text changed to the desired fonts.

12. Place the cursor on the "Move" option and press the **left mouse button**. The message **"Pick region to be moved—press Cancel to quit"** will be displayed in the bottom left corner of the screen.

13. Place the cursor near the text to be moved and press the **left mouse button**. Drag the cursor until the box surrounds the text and press the **left mouse button** again. The message **"Pick reference point—press CANCEL to quit"** will be displayed in the bottom left corner of the screen.

14. Place the cursor on the text and press the **left mouse button**. Position the box at a new location where the text can be moved and press the **left mouse button**. Verify that text has moved on the screen. Press the **right mouse button** twice to terminate the "Move" option.

15. Place the cursor on the "Copy" option and press the **left mouse button**. The message **"Pick region to be copied—press CANCEL to quit"** will be displayed in the bottom left corner of the screen.

16. Place the cursor near the text to be copied and press the **left mouse button**. Drag the cursor until the box surrounds the text and press the **left mouse button** again. The message **"Pick reference point—press CANCEL to quit"** will be displayed in the bottom left corner of the screen.

17. Place the cursor on the text and press the **left mouse button**. Position the box at a new location where the text can be copied and press the **left mouse button**. Verify that text was copied on the screen. Press the **right mouse button** twice to terminate the "Copy" option.

18. Place the cursor on the "Erase" option and press the **left mouse button**. The message **"Pick region to be deleted"** will be displayed in the bottom left corner of the screen.

19. Place the cursor near the text that was copied and press the **left mouse button**. Drag the cursor until the box surrounds the text and press the **left mouse button** again. The message **"Delete this region? left = YES, Right = NO"** will be displayed in the bottom left corner of the screen.

20. Press the **left mouse button**. Verify that the text on the screen was deleted. Repeat Step 19 to delete all the text that was typed in Step 10. When completed, press the **right mouse button** to cancel the "Erase" option.

21. Return to the *TEXT* menu. Place the cursor the "EDIT" option and press the **left mouse button** to access the *EDIT TXT* menu.

22. Place the cursor on the "Color" option and press the **left mouse button**. The message **"Box the text to be changed"** will be displayed in the bottom left corner of the screen.

23. Place the cursor near the text to be changed and press the **left mouse button**. Drag the cursor until the box surrounds the text and press the **left mouse button** again. The message **"Pick the new text color from the color bar"** will be displayed in the bottom left corner of the screen.

24. Place the cursor on one of the colors from the *Color Bar* menu and press the **left mouse button**. Verify that the text changed to the selected color. Press the **right mouse button** to terminate the "Color" option.

25. Repeat Steps 22 through 24 for the remaining colors.

26. Place the cursor on the "Size" option and press the **left mouse button**. The message **"Box the text to be changed"** will be displayed in the bottom left corner of the screen.

27. Place the cursor near the text to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the text and press the **left mouse button** again. The message **"Enter the new text size >"** will be displayed in the bottom left corner of the screen.

28. Type a text size of your choice and press the **left mouse button**. Verify that the size of the selected text was changed.

29. Place the cursor on the "Just" option and press the **left mouse button**. The message **"Box the text to be changed"** will be displayed in the bottom left corner of the screen.

30. Place the cursor near the text to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the desired text and press the **left mouse button** again. The message **"Enter text justification - ('L'=Left, 'R'=Right, 'C'=Center) >"** will be displayed in the bottom left corner of the screen.

31. Type **R** and press **<enter>**. Verify that the selected text on the screen has changed to right justification.

32. Type **L** and press **<enter>**. Verify that the selected text on the screen has changed to left justification.

33. Type **C** and press **<enter>**. Verify that the selected text on the screen has changed to center justification. Press the **right mouse button** to terminate the "Just" option.

34. Place the cursor on the "FONT" option and press the **left mouse button** to access the *Font* menu. Place the cursor at one of the desired fonts and press the **left mouse button**. The message **"Box the text to be changed"** will be displayed in the bottom left corner of the screen.

35. Place the cursor near the text to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the desired text and press the **left mouse button** again. Verify that the text changed to the desired font. Press the **right mouse button** to terminate the "FONT" option. Press the **right mouse button** again to cancel the *TEXT* menu.

36. Return to the *Primary Event Tree Graphics* menu. Place the cursor on the "Exit" option and press **<enter>**. The message **"ARE YOU SURE? Enter Y to Quit anyway"** will be displayed in the bottom left corner of the screen. Type **Y** to access the *Event Tree Graphics System* menu.

37. IF: Steps 1 through 36 were performed successfully,

    THEN: GO TO Step 38.

    ELSE: Record any step(s) that was unsuccessful. Generate an anomaly report.

38. End of Section 4.0

C-3. FEP NONVITAL FEATURES: PROCEDURE 3—SECTION 4

1. Place the cursor on the "P&ID Editor" option and press **<enter>** to access the *Primary P&ID Graphics* menu.

2. Place the cursor on the "FILE" option and press the **left mouse button** to access the *FILE* menu. Place the cursor on the "Load" option and press the **left mouse button.** The message **"Enter file name >"** will be displayed at the bottom of the screen. Type **Cont** and press **<enter>**. Verify that the P&ID diagram is displayed on the screen.

3. From the *Primary P&ID Graphics* menu, place the cursor on the "FILE" option and press the **left mouse button** to access the *FILE* menu. Place the cursor on the "+Tran" option and press the **left mouse button.** Verify that a triangle symbol appears on the screen.

4. Place the triangle at a convenient location on the screen and press the **left mouse button.** The message **"Enter Transfer file name >"** will be displayed in the bottom left corner of the screen. Type the file name **symbol** and press **<enter>**. Verify the triangle with the SYMBOL name on the screen. Press the **right mouse button** to cancel the "+Tran" option.

5. Place the cursor on the "Save" option. The message **"Enter file name or CR for file. [CONT.PID] >"** will be displayed in the bottom left corner of the screen. Press the **left mouse button** again and the message **"File already exists. Replace file? <Y?N>"** will be displayed in the bottom left corner of the screen. Type **Y** and press **<enter>**.

6. From the *FILE* menu, place the cursor on the "Xfer→" option and press the **left mouse button.** The message **"Pick transfer symbol to transfer"** will be displayed in the bottom left corner of the screen. Place the cursor on the triangle symbol and press the **left mouse button.** Verify on the screen that the symbol transferred.

7. From the *Primary P&ID Graphics* menu, place the cursor on the "FILE" option and press the **left mouse button** to access the *FILE* menu. Place the cursor on the "←Xfer" option and press the **left mouse button.** Verify that a list of file names are displayed on the screen. The message **"Pick the file to transfer to"** will be displayed in the bottom left corner of the screen.

8. Place the cursor on the file named CONT and press the **left mouse button.** Verify that the P&ID is displayed on the screen.

9. Place the cursor on the "EDIT" option and press the **left mouse button** to access the *EDIT* menu. Place the cursor on the "ATTRIB" option and press the **left mouse button** to access the *ATTRIBUTES* menu.

10. Place the cursor on the "Fill col" option and press the **left mouse button.** The message **"Pick a new color from the color bar"** will be displayed in the bottom left corner of the screen.

11. Place the cursor on a color from the *Color Bar* menu and press the **left mouse button.** The message **"Box the symbols to be changed"** will be displayed in the bottom left corner of the screen.

12. Place the cursor near one of the symbols to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the symbol and press the **left mouse button** again. Verify that the symbol changed to the selected color. Press the **right mouse button** to cancel the "Fill col" option.

13. Repeat steps 11 and 12 for the remaining colors.

14. From the *ATTRIBUTES* menu, place the cursor on the "Name Col" option and press the **left mouse button**. The message **"Pick a new color from the color bar"** will be displayed at the bottom on the screen.

15. Place the cursor on a color from the *Color Bar* menu and press the left **mouse button**. The message **"Box the symbols to be changed"** will be displayed in the bottom left corner of the screen.

16. Place the cursor near one of the symbols to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the symbol and press the **left mouse button** again. Verify that the symbol name changed to the selected color. Press the **right mouse button** to cancel the "Name col" option.

17. Repeat steps 15 and 16 for the remaining colors.

18. From the *ATTRIBUTES* menu, place the cursor on the "Line col" option and press the **left mouse button**. The message **"Pick a new color from the color bar"** will be displayed at the bottom of the screen.

19. Place the cursor on a color located on the *Color Bar* menu and press the **left mouse button**. The message **"Place a box around the lines to be changed—Press CANCEL to quit"** will be displayed in the bottom left corner of the screen.

20. Place the cursor near the line to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the line and press the **left mouse button** again. Verify that the line changes to the selected color. Press the **right mouse button** to cancel the "Line col" option.

21. Repeat steps 19 and 20 for the remaining colors.

22. From the *ATTRIBUTES* menu, place the cursor on the "Name size" option and press the **left mouse button**. The message **"Enter new text size"** will be displayed in the bottom left corner of the screen.

23. Type a symbol *name size of your choice* and press **<enter>**. The message **"Box the symbols to be changed"** will be displayed in the bottom left corner of the screen.

24. Place the cursor near one of the symbols to be modified and press the **left mouse button**. Drag the cursor until the box surrounds the symbol and press the **left mouse button** again. Verify on the screen that the symbol name size was modified. Press the **right mouse button** to cancel the "Name Size" option.

25. From the *ATTRIBUTES* menu, place the cursor on the "Name FONT" option and press the **left mouse button** to access the *FONT* menu. Place the cursor on one of the fonts and press the **left mouse button.** The message **"Box the symbols to be changed"** will be displayed in the bottom left corner of the screen.

26. Place the cursor near one of the symbols to be modified and press the **left mouse button.** Drag the cursor until the box surrounds the symbol and press the **left mouse button** again. Verify on the screen that the symbol name changed to the selected font. Press the **right mouse button** to cancel the "Name Font" option.

27. Place the cursor on the "EXIT" option and press the **left mouse button.** The message **"Current drawing not saved! Exit anyway? <Y?N>"** will be displayed on the bottom left corner of the screen. Type **Y** to return to the *FEP* main menu.

28. IF: Steps 1 through 27 were performed successfully

    THEN: Go to Step 29.

    ELSE: Record any step(s) that was unsuccessful. Generate an anomaly report.

29. End of Section 4.

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–6116
INEL-94/0039
Vol. 9

2. TITLE AND SUBTITLE

Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 5.0

Verification and Validation (V&V) Manual

3. DATE REPORT PUBLISHED

MONTH	YEAR
March	1995

4. FIN OR GRANT NUMBER

L2483

5. AUTHOR(S)

J. L. Jones, M. B. Calley, E. L. Capps, S. L. Zeigler,
W. J. Galyean, S. D. Novack, C. L. Smith, L. M. Wolfram

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 Engineering Laboratory
Lockheed Idaho Technologies Company
P.O. Box 1625
Idaho Falls, Idaho 83415

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 Safety Issue Resolution
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555-0001

10. SUPPLEMENTARY NOTES

11. ABSTRACT *(200 words or less)*

A verification and validation (V&V) process has been performed for the System Analysis Programs for Hands–on Integrated Reliability Evaluation (SAPHIRE) Version 5.0. SAPHIRE is a set of four computer programs that the Nuclear Regulatory Commission has developed for the performance of probabilistic risk assessments. These programs allow an analyst to perform many of the functions necessary to create, quantify, and evaluate the risk associated with a facility or process being analyzed. The programs included in this set are Integrated Reliability and Risk Analysis System (IRRAS), System Analysis and Risk Assessment (SARA), Models and Results Database (MAR–D), and Fault Tree/Event Tree/Piping and Instrumentation Diagram (FEP) graphical editor. The intent of this program is V&V of successive versions of SAPHIRE. The SAPHIRE 5.0 V&V plan is based on the SAPHIRE 4.0 V&V plan with revisions to incorporate lessons learned from the previous effort. The SAPHIRE 5.0 vital and nonvital test procedures are based on the test procedures from SAPHIRE 4.0 with revisions to include the new SAPHIRE 5.0. The majority of the results from the testing was acceptable; however, some discrepancies between expected code operation and actual code operation were identified. Modifications that have been made to SAPHIRE are identified.

12. KEY WORDS/DESCRIPTORS *(List words or phrases that will assist researchers in locating the report.)*

Probabilistic Risk Assessment
IRRAS
SAPHIRE

13. AVAILABILITY STATEMENT

Unlimited

14. SECURITY CLASSIFICATION

(This Page)
Unclassified

(This Report)
Unclassified

15. NUMBER OF PAGES

16. PRICE