BROOKHAVEN
NATIONAL LABORATORY

BNL-NUREG-72381-2004-CP

# *Issues Associated with Probabilistic Failure Modeling of Digital Systems*

**T. L. Chu, G. Martinez-Guridi, J. Lehner**
Brookhaven National Laboratory
Building 475C, P.O. Box 5000
Upton, New York 11973
Chu@bnl.gov, Martinez@bnl.gov, Lehner@bnl.gov

**D. Overland**
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
DHO1@nrc.gov

*Presented at the 4th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies*
Columbus, Ohio
September 19-22, 2004

June 2004

**Energy Sciences and Technology Department**

**Brookhaven National Laboratory**
P.O. Box 5000
Upton, NY 11973-5000
www.bnl.gov

*Managed by*
Brookhaven Science Associates, LLC
for the United States Department of Energy under
Contract No. DE-AC02-98CH10886

BNL-NUREG-72381-2004-CP
Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies (NPIC&HMIT 2004), Columbus, Ohio, September, 2004

# ISSUES ASSOCIATED WITH PROBABILISTIC FAILURE MODELING OF DIGITAL SYSTEMS*

**T. L. Chu, G. Martinez-Guridi, and J. Lehner**
Brookhaven National Laboratory
Building 475C, P. O. Box 5000
Upton, New York 11973
Chu@bnl.gov, Martinez@bnl.gov

**D. Overland**
U. S. Nuclear Regulatory Commission
Rockville, Maryland 20852-2738
DHO1@nrc.gov

**Keywords**: Probabilistic Risk Assessment, Digital Systems, Safety-Critical Systems

## ABSTRACT

The current U.S. Nuclear Regulatory Commission (NRC) licensing process of instrumentation and control (I&C) systems is based on deterministic requirements, e.g., single failure criteria, and defense in depth and diversity. Probabilistic considerations can be used as supplements to the deterministic process. The National Research Council has recommended development of methods for estimating failure probabilities of digital systems, including commercial off-the-shelf (COTS) equipment, for use in probabilistic risk assessment (PRA). NRC staff has developed informal qualitative and quantitative requirements for PRA modeling of digital systems.

Brookhaven National Laboratory (BNL) has performed a review of the-state-of-the-art of the methods and tools that can potentially be used to model digital systems. The objectives of this paper are to summarize the review, discuss the issues associated with probabilistic modeling of digital systems, and identify potential areas of research that would enhance the state of the art toward a satisfactory modeling method that could be integrated with a typical probabilistic risk assessment.

## 1. INTRODUCTION

At the request of the U.S. NRC, the National Research Council formed a committee to conduct a study on application of digital I&C technology to commercial nuclear power plant (NPP) operations (USNRC, 1997). In the study, the committee investigated the important safety and reliability issues and provided recommendations on the issues. On the issue associated with safety and reliability modeling methods, the recommendations include: (1) development of methods for estimating the failure probabilities of digital systems for use in PRA, (2) inclusion of the relative influence of software failure on system reliability in PRAs of systems containing digital components, (3) development of expertise to understand the requirements for gaining confidence in digital systems and the limitations of quantitative assessment, and (4) development of advanced techniques for analysis of digital systems that would increase the confidence and reduce the uncertainty in quantitative assessments.

* This work was performed under the auspices of the U.S. Nuclear Regulatory Commission.

In 2000, NRC started a research plan on digital I&C. NRC staff prepared a paper on "What PRA Needs From a Digital Systems Analysis," (Arndt, 2002) which specifies the qualitative and quantitative modeling requirements of digital systems, including compatibility with existing PRAs, level of detail of internal structure, modeling of dependency, modeling of software failures, and diversity of digital systems. The research subjects related to PRA modeling include software requirement specification, operating experience, survey of reliability methods, fault injection and Markov modeling, and software reliability modeling. In addition, BNL has performed a literature review on issues associated with probabilistic failure modeling of digital systems, a review of the current NRC guidance and regulation associated with reliability modeling of digital systems, a review of existing methods and tools for modeling digital systems, an failure mode and effects analysis (FMEA) of a hypothetical reactor protection system (RPS) based on the Tricon platform, and a review of the existing failure databases of digital components.

The objectives of this paper are to summarize the issues associated with probabilistic failure modeling of digital systems, review the methods for modeling digital systems, and provide suggestions on additional research that should be performed in developing a commonly acceptable method for modeling digital system.

## 2. ISSUES ASSOCIATED WITH FAILURE MODELING OF DIGITAL SYSTEMS

A PRA is an integrated model of a NPP, and digital systems at the plant have extensive interfaces with the rest of the plant. Therefore, it is important that the interfaces between the digital systems and the rest of the plant be properly accounted for. Adequate supporting analyses, e.g., FMEA, have to be performed to determine the impacts of digital systems. For a digital control system that is normally operating, e.g., feedwater control system, its failure could cause an initiating event, e.g., loss of feedwater. Therefore, a model that estimates the frequency of such an initiating event is needed. For a protection system, e.g., RPS and engineered safety features actuation system (ESFAS), a model that estimates the probability that the system would fail to perform its function for different initiating events and different accident sequences is needed. Spurious actuation of these systems could also contribute to initiating events. For instrumentation systems, e.g., indications of physical conditions of the plant, a model for estimating the probability of incorrect indication or spurious actuation is needed.

A realistic model of a digital system should be able to capture the unique features of the system. The unique features may include software, diagnostics, self-correction, signal validation, synchronization, and unique communication means, e.g., buses, LAN, and fiber optic connections. Fault tolerant features tend to improve the reliability of digital systems and should be accounted for in the model. Digital systems have unique failure modes and causes that could affect the system adversely and should also be accounted for in the model. For example, software failures are known to be dominated by errors associated with requirement specification (Hecht, 2001a) and are potential common cause failures (CCFs) of a redundant system. Unique features of digital systems could be implicitly captured in the data used in a model, or explicitly modeled, depending on the level of detail of the model. In either case, good applicable data has to be used.

### 2.1 Are Software Failure Rates Meaningful?

Modeling of software failures in terms of failure rates or equivalently mean times to failure has been very common, e.g., reference (Lyu, 1996). The methods for estimating software failure rates use test data and are extensions of hardware reliability methods to software. In testing hardware, e.g., starting of a pump, identical tests are performed, and the results are used to estimate the failure probability. In software testing, different samples from the input domain of the software are taken as input to the software, bugs are fixed when they are identified, and the test results are used to estimate failure rates in the same way the hardware test results are used. Reliability growth methods use the data collected, usually in the form of successive execution times between failures, to estimate current reliability and predict future reliability growth. Due to the high reliability of the safety-critical software, a very large number of tests would be needed to obtain high confidence of the low failure probabilities (Littlewood, 1991).

Some experts (Leveson, 1991; Singpuwalla, 1995) have the opinion that software is deterministic, i.e., given the same input, it will always produce the same output, and it may not be meaningful trying to model it probabilistically. Some even argue that software does not fail, because they always do what they are designed to do. Unlike hardware, which may fail due to physical degradation of the components themselves, software does not age. Either a software fault exists at the beginning of life, or it doesn't. It does not come into existence at some point in time. This argues against the use of aleatory models for random failures that quantify the fraction of times the software fails. Instead, it implies an epistemic model in which the software is always failed (with some probability) or always good (with the complementary probability). That is, we do not know if the software would fail, and represent our knowledge about the software failure in terms of a probability.

The "error forcing context" (EFC) concept of (Garrett, 1999) is consistent with the idea that software is deterministic. The Dynamic Flowgraph Methodology (DFM) is used to identify the EFCs in the form of fault tree prime implicants. The prime implicants generated using the DFM method are equivalent to the cutsets generated by standard fault tree method, except that they represent a more detailed model of the system including explicit modeling of timing and software. The quantification of software failures would involve a quantification of the likelihood of the EFCs in the form of prime implicants. Unlike hardware failures, the EFCs are external to the software and do not represent any changes/degradations of the software. This is similar to the "EFC" concept of human reliability analysis (USNRC, 2000), where the EFCs increase the likelihood of human diagnosis errors. Having found the EFCs for software, one would like to correct them by fixing the software bugs, unless the likelihood of the error-forcing context is low enough not to justify fixing.

The "EFCs" concept appears to contradict the typical assumption of fault tree analysis, that the basic events are independent and the same basic event representing a software failure is applicable to many different boundary conditions or scenarios defined in the event trees. In reality, the context or boundary condition defined in a PRA is never detailed enough to specify a single input point from the domain of the software. Instead, a region in the input domain is used and it may contain the corner of the domain that the software is not designed for. The software failure probability used in a fault tree can be considered as the probability that the region of the domain contains the unknown EFC.

The EFCs of DFM can be considered the dark corners of the input space of the software that the software designer did not take into consideration. The identification and correction of the EFCs would eliminate these dark corners. An obvious question is whether or not DFM would identify all dark corners and make the software perfect. The answer is probably no. We could consider DFM as another method for checking/debugging software. Similarly, software testing and other methods of software hazard analysis would identify and eliminate some dark corners. Identification and elimination of the dark corners lead to a reduction in the likelihood that any of the remaining dark corners would be triggered. It is the realization of the dark corners of the input space that causes software to fail. The realization depends on the operating environment of the software including the hardware it runs on and its input, and is aleatory.

## 2.2 How Diverse Is Diverse Enough?

Defense in depth and diversity (D3) is a very important consideration of the NRC regulation of digital systems. For a replacement of the RPS or ESFAS, a D3 analysis is required. In a D3 analysis, CCFs are postulated one at a time, and diversity has to be demonstrated for each of the accidents in the safety analysis report (Preckshot, 1994). The assumption that the CCFs occur without considering their likelihood is very conservative and is a situation which PRA considerations could help. In PRA space, diversity would mean that the systems or components would fail independently, i.e., their failure probabilities can be multiplied. A question is how diverse is diverse enough to do so.

Use of the same microprocessor and operating system is an example of the factors that may contribute to CCF of digital systems. Both inter-system and intra-system CCFs should be considered. For example, RPS and ATWS mitigation system are required to be diverse. Systems manufactured by different companies are often considered diverse. What if they use the same CPU? Often, a safety critical digital system consists of channels with identical hardware and software. CCF of both hardware and software has to be considered. In the case of hardware, in general, standard CCF methods could be used, and the hardware made by different manufacturers can probably be considered diverse. For identical software in identical channels, complete dependence has to be assumed. The defect found in the sequencer logic of the Turkey Point NPP (FPL, 1994) could affect all four sequencers at the plant and is an example of such type of CCF.

Diversity of software is difficulty to demonstrate. Knight and Leveson (Knight, 1986) performed an experiment using 27 versions of a program, and the results show that the independence assumption is rejected with 99% confidence. The dependence came from programmers making equivalent logic errors.

## 2.3.1 Adequacy of Modeling and Analysis Methods

Different modeling and analysis methods serve different purposes. Some methods are supporting analyses that would identify different ways digital systems could fail which provides information on how the system should be modeled, e.g., FMEA and hazard analysis, or verify the assumption used in modeling, e.g., independence assumption. Some represent models of the behavior of digital systems, e.g., fault injection simulation and Petri net. They are useful tools for evaluating the design. Some are probabilistic failure models, e.g., fault tree and Markov model. Some methods are used in performing quantitative assessment of software reliability. These modeling and analysis methods are evaluated below based on the previously discussed requirements and issues.

## 2.3.1 Supporting Analysis Methods

Probabilistic modeling of digital systems requires that dependencies be modeled properly. In particular, synchronization, voting, data communication are physical interactions between processors and redundant channels, and can potentially introduce dependent failures. The incident at Southern California Edison in which communication failure caused loss of both primary and backup security systems (Hecht, 2001b) is an example of dependent failures caused by communication between redundant systems. It is desirable that deterministic evaluations be performed to develop guidance on how such dependencies should be modeled.

In the life cycle of a digital system, many activities/analyses take place in an attempt to make sure that the system is free from faults. The information gathered and the results of the analysis are essential to probabilistic failure modeling. For example, IEEE Standard 7-4.3.2-2003 (IEEE, 2003) has an annex that discusses software hazard analysis and identification of abnormal conditions and events. The methods discussed include preliminary hazard analysis, fault tree analysis, FMEA, system modeling, software requirement hazard analysis, design reviews and code reviews, and simulator/plant model testing. Similar to other guidance documents, the methods are only briefly described. They are also briefly explained with simple examples provided in textbooks (Lyu, 1996; Leveson, 1995). The concepts of the methods are simple but the quality of their application depends on how carefully they are carried out, the level of detail of the analysis, the availability of detail information, the qualification and experience of the analysis team, and the resource limitations. For important systems, extensive analyses may have been performed, but the analyses are typically proprietary and not available to the public. On the other hand, it is commonly believed that it is not possible to prove that software of moderate or high complexity is fault free.

### 2.3.2 Fault Injection Simulation

Fault injection method has been used to validate the fault handling mechanisms of systems as well as to provide a model for system-level manifestation of faults. The University of Virginia (UVA) has developed a generic processor fault model of digital processors to the level of individual bits (Cutright, 2003). The method performs fault injection experiments on the model of a processor including the software that runs on it and determines the coverage of the processor, which is then used as a parameter of the Markov model of the processor. In general, the approach can also be extended to model a system with multiple processors.

The fault injection method appears to be a good approach for modeling the behavior of a digital processor, and can be used to evaluate its design. The UVA model can probably be used to support a PRA in providing an estimate of fault coverage of a processor. An important issue is whether or not all possible failures of a processor manifest themselves in the form of a single "stuck-at" type of faults. The model's ability to capture software failures is limited due to its limited variability in the input to the software. Whether or not an integrated simulation model of a digital system, e.g., a reactor protection system, would generate meaningful reliability results, i.e., probability of failure on demand, remains to be evaluated.

### 2.3.3 Petri-Net

The Petri-net method has been used as a modeling method for the behavior of software. The advantage of Petri-net is its ease of modeling the behavior of a dynamic system. A Petri-net model can be analyzed to show the presence or absence of safety properties, such as hazardous conditions, system deadlock, or unreachable states. The method has been used as a tool for an FMEA (Goddard, 1996) to identify failures and their effects. It has not been commonly used by the nuclear industry but has been used for reliability modeling of computer-based systems (Malhotra, 1995). Stochastic Petri-net is a Petri-net whose time of firing is exponentially distributed. The model is similar to a Markov model and has to be converted into a Markov model in order to be solved.

Petri-net method has the capability of modeling the unique features of digital systems at a level higher than that of the fault injection method of UVA. Its use as a tool for FMEA and hazard analysis probably should be further explored. Its value as a probabilistic modeling method is probably limited by the limitations that apply to a Markov model.

### 2.3.4 Fault Tree Analysis

Fault trees and event trees are the basic logic structures of a PRA. Therefore, it is very desirable to model every relevant system, including digital systems, within this framework. Even if a more sophisticated model has to be used, it is desirable to convert the results of the analysis into a fault tree format. A few methods and analysis that can be considered variations of the standard fault tree methods are discussed here.

Parts Count and Part Stress Method - The military handbook on reliability prediction of electronic equipment MIL-HDBK-217F (RAC, 1991) contains two methods for estimating failure rates of boards/systems, the parts count method, and the part stress method. The methods are applicable to systems/channels with components in series, and any redundancy of a system has to be modeled using other methods, e.g., Markov model. They have been used mainly in the defense industry.

Traditional Fault Tree Method - AP600 (Westinghouse, 1996) is an advanced design that has been reviewed and approved by the NRC. It has an integrated digital I&C architecture. The fault tree models of the AP600 PRA follow the method of standard fault tree analysis. CCF was modeled for most of the digital components. It is not known how the CCF probabilities were estimated. The CCF probabilities for software failures do not seem to have a good basis. They range from 1E-5 to 1E-6 and are considered the goals of the design.

It is not clear how digital features, such as voting, synchronization, and data communication, are accounted for. The modeling of software CCF as basic events requires that a philosophical framework be developed. As discussed earlier, software failures are sensitive to the contexts or boundary conditions. Modeling them simply as basic events seems to contradict the concept, because the same software CCF event would be ANDed with many different combinations of basic events, which define many different contexts.

Dynamic Fault Tree - The word "dynamic" has been used in different applications to represent different meaning. In general, it represents a model being better able to account for timing of events. The dynamic fault tree method of Dugan (Dugan, 1992) is an extension of the standard fault tree method. It introduces special gates that handle the order in which events occur, e.g., a priority AND gate generates an output only if the inputs occur in a particular sequence. The method is a straightforward extension of the standard fault tree analysis method.

Dynamic Flowgraph Methodology - The DFM method (Garrett, 1999) is capable of modeling timing and software and has been proposed as a tool for safety analysis of digital systems. An important contribution of the paper is the concept of EFCs. The simple models of example systems in the papers were used to demonstrate the method. They demonstrated that the method is modeling digital systems at a level much higher than that of the fault injection simulators of UVA, i.e., only the algorithms of application software are modeled. Its use as a behavior model of application software for the purposes of identifying EFC is a reasonable application of the method, subject to the limitation of software complexity. Its use as a probabilistic failure analysis method, i.e., integration with a PRA, has not been demonstrated. It appears that such an application would require all possible contexts be identified and evaluated, which may not be realistically done. A realistic model would require that the world outside the system be modeled, e.g., deterministic and probabilistic model of the reactor coolant system.

### 2.3.5 Markov Model

Markov model is a well-established method for modeling systems and has been used by the process industry to model digital systems. It has also been used in modeling non-digital systems at NPPs, but its integration with existing PRA models may not be straight forward. It is suitable for modeling digital design features, such as fault detection, recovery, and reconfiguration, by assuming that software works perfectly. Its capability to capture the contribution of software failures to system reliability is limited.

Application of Markov model method to digital I&C systems depends on whether or not the model realistically represents the system, whether or not good data is available to support the quantification, and whether or not good physically meaningful reliability measures are derived. Realistic modeling depends on the supporting deterministic analysis to determine the failure modes and dependencies that have to be taken into consideration.

### 2.3.6 Quantification of Software Reliability

Current methods for quantitative assessment of software failure rates or probabilities require test results and sometimes, expert judgment. For example, Smidts and Li (Smidts, 2002) used the number of detected defects with the PIE method to estimate the probability of failure of a personnel access control system, and AIAA standard (AIAA, 1992) uses reliability growth models to determine the number of tests needed to reach the desired confidence on satisfying reliability requirements. PRISM (RAC, 1998) uses field data in a reliability growth model to estimate software failure rates. Dahll (Dahll, 2002) proposed estimating software failure probabilities using the Bayesian Belief Network (BBN) method which combines expert judgment with available data.

Due to inadequate data, the quantitative methods probably are not adequate to demonstrate the expected low failure probability/rate of safety-critical software. Their implementation into a PRA also requires a technical basis be established.

## 2.4 Adequacy of Failure Data

In general, a failure database should include the data needed for the specific modeling methods. For example, a database in support of a Markov model should contain data for all the transition rates of the model. In particular, it is desirable that the data needed to model unique features of a digital system be available. Different features may be implemented at different levels of detail in the design of a digital system and require data at different levels of detail.

BNL reviewed available information of the some of the available failure databases, including MIL-HDBK-217F (RAC, 1991), PRISM (RAC, 1998), Telcordia, NUREG/CR-6734 Volume 2 (Hecht, 2001b), and Government-Industry Data Exchange Program (GIDEP). The review seems to indicate the following weaknesses of the existing databases:

- The failure rates were estimated by grouping failure data of components from diverse sources, and the raw data and failure descriptions are generally not available. Therefore, fault tolerant features of the digital systems are built in the failure rate estimates. For example, a design with better cyclic redundant check (CRC) cannot be differentiated from one that has an inferior CRC.
- The failure rate estimates are not broken down into failure modes and causes. The level of detail at which failure rates were estimated and the completeness of the component types remain to be further evaluated for specific modeling methods.
- The reported failure events used in estimating failure rates are those that the systems were not able to diagnose and correct. That is, faults that are detected and corrected are not considered failures. This means no additional credit for detection and correction should be given to a model using the failure rates.
- Software failures are not adequately captured in the databases. Due to lack of event descriptions in the databases, experience of software CCF in redundant systems is in general not available.
- Diagnostics coverage is a parameter that depends on the specific design of the diagnostic software. It is probably difficult to collect generic estimates of diagnostic coverage.

## 3. CONCLUSIONS

Current methods, e.g., fault tree analysis and Markov model, were developed for hardware failures, are probably adequate for modeling hardware failures of digital systems and do not adequately capture software failures. No commonly accepted method to include software failures in a reliability model exists. A philosophical framework for software failures has to be developed to provide the basis of an acceptable method. It should address the issues on the meaningfulness of software failure rates and interactions between hardware and software. A method for modeling software failures that takes into consideration CCF of software as well as hardware is desirable.

Digital systems extensively use features, such as data communication, voting, and synchronization, which are physical interactions between redundant channels/ components. These features could introduce dependent failures among redundant channels. Supporting analysis should be performed to verify independence. Guidance on what is acceptable, i.e., the channels can be considered independent, should be developed. Guidance on how such dependencies should be modeled should be developed.

Modeling digital systems at the level of individual processors is needed to capture some important digital features, e.g., diagnostics and reconfiguration. Case studies should be performed to demonstrate the feasibility and capability of different methods, and identify the data needed to support the analyses.

The review of failure rate databases found that software failures are not adequately captured in the databases. Failure rates are not broken down into failure modes and causes. Fault tolerant features, such as CRC, are built in the failure rate estimates, and faults that are detected and corrected are not reported. They have a significant implication on how the rates should be used in modeling.

## REFERENCES

American National Standards Institute and American Institute of Aeronautics and Astronautics, "Recommended Practice for Software Reliability," ANSI/AIAA R-013-1992.

Arndt, S. A., Thornsbury, E. A., and Siu, N. O., "What PRA Needs from a Digital Systems Analysis," Proceedings of the 6th International Conference on Probabilistic Safety Assessment & Management, San Juan, Puerto Rico, June 2002.

Cutright, E., Delong, T., and Johnson, B., "Generic Processor Fault Model," University of Virginia, Technical Report UVA-CSCS-NSE-004, Revision 00, August 1, 2003.

Dahll, G., Gran, B.A., and Liwang, B., "Decision Support for Approval of Safety Critical Programmable Systems," NEA/CSNI/R(2002)1/VOL1.

Dugan, J. B., et al., "Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems," *IEEE Transactions on Reliability*, Vol. 41, N. 3, September 1992.

Florida Power and Light Company, Licensee Event Report, 94-005, 1994.

Garrett, C., and Apostolakis, G.A., "Context in the Risk Assessment of Digital Systems," *Risk Analysis*, Vol. 19, No. 1, 1999.

Goddard, P. L., "A Combined Analysis Approach to Assessing Requirements for Safety Critical Real-Time Control Systems," Hughes Aircraft Company, IEEE Proceedings, Annual Reliability Maintainability Symposium, 1996.

Hecht, M., and Hecht, H., "Digital Systems Software Requirement Guidelines, Guidelines", NUREG/CR-6734, volume 1, August 2001a.

Hecht, M., and Hecht, H., "Digital Systems Software Requirement Guidelines, Failure Description", NUREG/CR-6734, volume 2, August 2001b.

"IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations," IEEE Std. 7-4.3.2-2003.

Knight, J.C., and N. G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming," IEEE Transactions of *Software Engineering*, 12(1) 96-109, 1986.

Leveson, N., "Software Safety in Embedded Computer Systems", *Communications of the ACM*, 34, 34-46, February 1991.

Leveson, N.G., *Safeware, System Safety and Computers*, University of Washington, Addison-Wesley Publishing Company, 1995.

Littlewood, B., "Software Reliability Modeling: Achievements and Limitations," CompEuro '91. 'Advanced Computer Technology, Reliable Systems and Applications'. 5th Annual European Computer Conference, 13-16 May 1991.

Lyu, M. R., *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, 1996.

Malhotra, M., and Trivedi, K.S., "Dependability Modeling Using Petri-Net," *IEEE Transaction on Reliability*, Vol. 44, Issue 3, September 1995.

National Research Council, "Digital Instrumentation and Control Systems in Nuclear Power Plants, Safety and Reliability Issues," Final Report, Committee on Application of Digital Instrumentation and Control Systems to Nuclear Power Plant Operations and safety, National Research Council, National Academy Press, 1997.

Preckshot, G., "Method for Performing Diversity and Defense-in Depth Analyses of Reactor Protection Systems," Lawrence Livermore National Laboratory, NUREG/CR-6303, December 1994.

Reliability Analysis Center, "Reliability Prediction of Electronic Equipment," Rome Laboratory, MIL-HDBK-217F, December 1991.

Reliability Analysis Center, and Performance Technology, "New System Reliability Assessment Method," IITRI Project No. A06830, June 1, 1998.

Singpurwalla, N.D., "The Failure Rate of Software: Does It Exist?" *IEEE Transactions on Reliability*, Vol. 44, No. 3, September 1995.

Smidts, C., and Li, M., "Validation of a Methodology for Assessing Software Quality," Draft Complete on February 2002.

USNRC, "Technical Basis and Implementation Guidelines for A Technique for Human Event Analysis (ATHEANA), NUREG-1624, Rev. 1, May 2000.

Westinghouse, "AP600 Probabilistic Risk Assessment," Westinghouse Electric Corporation, ENEL, Revision 7, June 28, 1996.