Symposium on Nuclear Power Systems (SNPS'97), Nov. 11-14, 1997, Albuquerque, New Mexico

MEADEP and Its Application in Dependability Analysis for A Nuclear Power Plant Safety System

Dong Tang, Myron Hecht, Xuegao An
SoHaR Incorporated, Beverly Hills, California

Robert Brill
Nuclear Regulatory Commission, Washington DC

Abstract

Although there are several measurement and model based approaches to assessing the compliance of critical computing systems with reliability requirements, applying these approaches requires sophisticated data analysis and mathematical skills so that reliability engineers often hesitate to perform such a task. The need to develop cost effective, credible, and easy-to-use tools to reduce difficulties in performing such tasks has thus been apparent. This paper presents a tool of this kind — MEADEP. MEADEP integrates techniques in graphical user interface programming, database engineering, dependability modeling, and statistical/numerical analysis, and provides a user-friendly interface on Windows 95 for non-expert users. Use of MEADEP on failure data from measurements produces quantitative evaluations of dependability for critical systems, while greatly reducing requirements for specialized skills in data processing, statistical analysis, dependability modeling and model solution from the user. The application of MEADEP on safety systems is demonstrated by modeling dependability for a nuclear power plant safety system based on the Eagle 21 architecture and its early field failure reports.

1. Introduction

As safety and other critical systems in which software plays decision making and control roles are increasingly applied in the field such as nuclear power safety management and air traffic control, it becomes necessary to develop objective methods to assess the compliance of these systems with reliability requirements. In this regard, quantitative assessment of dependability[1] for critical digital systems is a vital issue. There are three general approaches to dependability evaluation for computing systems: in-process assessment, pre-deployment assessment, and in-field assessment [Neil96]. This classification can be mapped to the three phases of dependability evaluation defined in [Iyer96]: design phase, prototype phase, and operational phase.

In the design phase, a system is typically modeled by using probabilistic models [Laprie84, Scott87, Arlat90] or simulation methods [Goswami93, Tausworthe96]. This approach relies on component level failure rates published in handbooks or supplied by manufacturers. The approach provides an early indication of system dependability, but many assumptions have to be made in order to build analytical or simulation models. These assumptions as well as the underlying parameters later need to be validated by actual measurements.

In the prototype phase, assessment is typically done by product testing and reliability growth modeling [Musa87]. This approach involves fewer assumptions than the first, but it can address the reliability growth only for software with failure rates from 10^{-1} to 10^{-5} per hour [Butler93]. Reliability growth models do not furnish creditable predictions when there are few observed failures, as must be the case for safety systems. In addition, these models lack ability to account for the internal structure of a complex system with hardware, software, redundancy provisions and fault recovery processes.

Measurement-based dependability evaluation [Iyer96, Tang95a] is well suited for the operational phase. In many cases, it is possible to use a combination of measurement and dependability models to develop a quantitative assessment as shown in [Hsueh88, Lee93, Tang95]. Early studies of DEC [Castillo82] and IBM [Iyer85] operating system failures found that there is a strong correlation between system workload and software failures. Further research on mature fault tolerant real-time systems showed that residual software faults lead to a failure behavior which can be characterized by a failure rate and a certain failure arrival distribution [Nagle82, Adams84, Hsueh88] and that a majority of such failures could be masked (without obvious impact on applications) by the use of physical redundancy [Gray90, Lee95, Tang95]. These results provide a basis for modeling software failures as a stochastic process for real-time and fault tolerant systems.

Not only can the measurement-based dependability evaluation approach be used in the system operational phase, it can also been applied to the late testing phase of a software system as demonstrated in [Tang95]. The primary advantage of this approach lies in use of measurements and models for interpretation of the measurements. Based on measurements, the approach produces various dependability measures (MTBF, availability, etc.) with stated confidence levels. The measurement-based dependability evaluation methodology developed in [Tang96] consists of feasible methods in data collection and processing, statistical analysis, and dependability modeling. It not only evaluates system dependability based on a statistically significant number of failures, but it also evaluates system dependability lower bounds at a specified confidence level where failures were rare.

However, applying measurement-based dependability evaluation approach involves difficulties in data processing, parameter estimation, model specification, and appropriate mapping from data to models. It is costly and time consuming for reliability engineers to overcome these difficulties. The need to develop tools that can reduce these difficulties has thus been apparent. In this paper, we introduce such a tool, MEADEP (MEAsure DEPendability), and demonstrate its application in modeling and analyzing dependability for a nuclear power plant's safety system. MEADEP integrates techniques in graphical user interface (GUI) programming, database engineering, dependability modeling, and statistical/numerical analysis, to provide non-expert users an easy-to-operate environment for producing dependability assessments for

[1] The *dependability* concept was proposed in the *15th International Symposium on Fault-Tolerant Computing* (FTCS-15) [Laprie85] and revised in FTCS-25 [Laprie95]. Dependability is defined as the "property of a computer system such that reliance can justifiably be placed on the service it delivers." Major measures of dependability include *reliability*, *availability*, *safety*, and *maintainability*.

real systems. Use of the tool on failure data from measurements produces quantitative assessments of dependability for critical systems, while greatly reducing requirements for specialized skills in data processing, statistical analysis, and dependability modeling from the user. Because MEADEP facilitates the use of measurement-based dependability analysis methods and reduces the cost of such analyses by providing various data processing and dependability analysis functions, it can become an integral part of engineering projects where dependability is an important consideration.

2. Overview of MEADEP

MEADEP is a failure data based dependability analysis and modeling tool. Dependability measures generated by MEADEP are either directly obtained from data, such as failure rate and event distribution, or evaluated by combined use of failure data and dependability models, such as system level reliability and availability. Thus two basic types of input to MEADEP are:

- Data — Structured failure reports containing information on failure time, location, impact and other failure characteristics
- Models — Graphical specifications of dependability models including reliability blocks and Markov chains

The output of MEADEP consists of results obtained from data and results evaluated from models where model parameters were estimated from data or given by users.

Results obtained from data include:

- Pie chart for event distribution
- Progressive curves over time for Mean Time Between Events (MTBE) and its confidence interval
- Histogram for Time Between Events (TBE) and Time To Recovery (TTR), with super-plotting of typical analytical functions, accompanied by the results of their goodness-of-fit tests
- The mean, lower and upper bounds for failure rate, recovery rate, and coverage

Results evaluated from models include:

- Mean Time Between Failures (MTBF)
- Reliability for a given time period
- Steady-state availability

The major functions of MEADEP are: data processing and editing, parameter estimation, graphical data analysis, graphical model generation and model solution. Particularly, MEADEP has the following features:

Support for data conversion: Structured data in a variety of formats (ASCII Delimited Text, Access®, dBASE®, Paradox®, etc.) can be converted to the MEADEP data format.

Estimation of parameters from data: Typically used parameters (failure rate, coverage, etc.) and their upper and lower bounds at a certain level of confidence are estimated by statistical routines taken from mature numerical libraries.

Graphical presentations of data: A number of graphical formats are provided to display dependability characteristics for data and results evaluated from models.

Graphical Input of models: A graphical “drag and drop” interface allows the user to create models hierarchically out of reliability block diagrams (including the k-out-of-n block) and Markov reward models [Goyal87].

Parametric analysis in solution: The model solution part of MEADEP allows a model to be run with a range of user-specified values for a selected parameter including time. The results can be displayed as a curve.

A library of dependability models: A library of dependability models, including primitive models for typical fault-tolerant architectures and complex models for real critical systems are included in MEADEP for reuse by users.

User friendly interface: For all of its functions, MEADEP provides a user-friendly GUI featuring menus, dialogs, pictures, printing previews, and extensive on-line help information.

MEADEP was developed on Windows 95 using Microsoft Visual C++, the Open Database Connectivity interface, the IMSL® Numerical libraries, and the Olectra® Chart graphical package. The parameter estimation methods used were based on [Kececioglu93, Tang95], and the model solution methods used were based on [Trivedi82, Reibman88].

3. Application of MEADEP on a Nuclear Power Plant Safety System

In critical applications, there are two broad categories of digital systems: (1) continuously operating real-time systems, and (2) on-line protection systems. The operational profiles radically differ for the two categories: continuous input (workload may fluctuate) for the first and intermittent input (rare events) for the second. The first category requires high availability and can tolerate component-level failures by redundancy provisions. Computer operating systems, air traffic control plant process control systems all fall into this category. Most previous work on measurement-based dependability evaluation has been for systems in this category as reviewed in [Iyer96]. The second category requires successful responses to emergent demands and a failed response can result in the loss of life and property. The nuclear power plant’s safety system is a typical representative of this category. MEADEP has been applied to the first category in evaluating operational availability for two air traffic control systems [Tang97]. This section demonstrates the MEADEP application on the second category by analyzing the sensitivity of the plant Mean Time Between Hazards (MTBH) to key parameters for a nuclear power plant’s safety system.

3.1 Dependability Modeling

The modeled configuration has two major components: a plant and a digital safety system which protects the plant by responding to and processing challenges. One of the safety systems installed in the plant studied was the Eagle 21 [Westinghouse91] digital safety system. A 3-level hierarchical model was developed for this configuration where levels 2 and 3 were based on the architecture of Eagle 21. Figure 1 shows the top-level plant model which reflects the intermittent operating profile.[2] Figure 2 shows the middle-level model, a safety system which consists of four channels working on a basis of 2-out-of-4 votes for a reactor trip (shut down reactor). Figure 3 shows the bottom-level model, a single channel which consists of four components. In this analysis, channel failures are assumed to be of the Byzantine type[3] because this type is the worst case failure mode and is hazardous to the protection function. The

[2] The heavy frame in this diagram means parameters λss and μss are evaluated from the lower level model SafSys. Similar for other diagrams.

[3] The faulty channel continues execution and lies when asked for information [Siewiorek92].

notation used in these figures is explained in Appendix A.

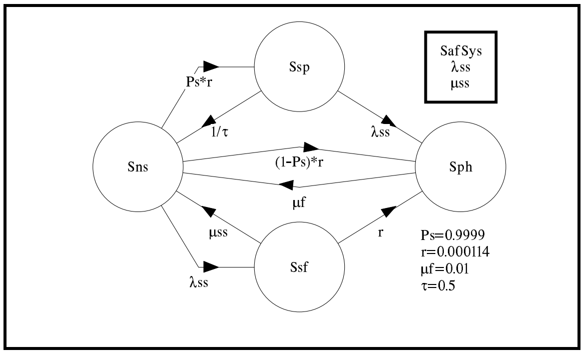

Figure 1 The Nuclear Plant Model

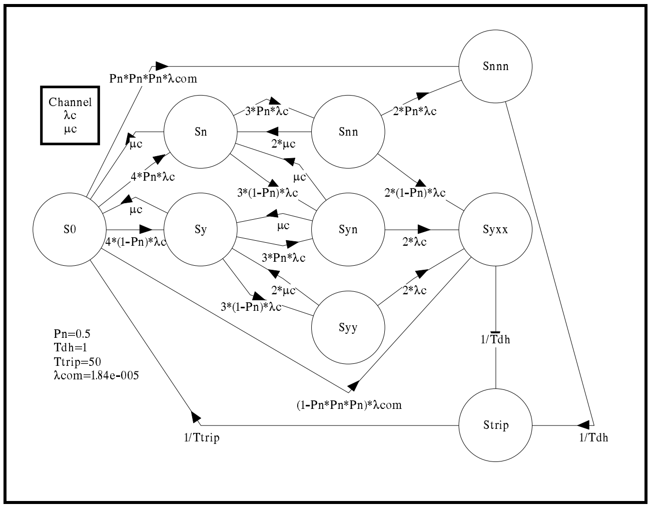

Figure 2 The Safety System Model (SafSys)

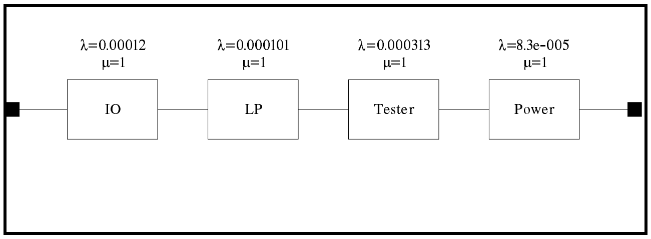

Figure 3 The Safety Channel Model (Channel)

In Figure 1, if a challenge arrives in the normal/safe state, the safety system will respond to it successfully with probability P_S and go to the safety processing state S_{sp} (modeled by transition P_S*r, from S_{ns} to S_{ps}). During the safety processing, if the safety system fails due to random failures, the plant will be in the hazard state (transition λ_{SS}, from S_{sp} to S_{ph}). Otherwise, the safety system will go back to the normal/safe state after the mean processing time τ (transition $1/\tau$, from S_{sp} to S_{ns}). When a challenge arrives in the normal/safe state, the safety system may respond to it unsuccessfully due to hardware/software design or implementation problems and go to the plant hazard state (transition $(1-P_S)*r$, from S_{ns} to S_{ph}).[4] Thus, maximizing P_S is the major goal for this model. Sometimes the safety system random failures occur in the normal/safe state and enters the safety failure state S_{sf} (transition λ_{SS}, from S_{ns} to S_{sf}).[5] The safety system will go back to the normal/safe state when the safety system failure is detected and handled (transition μ_{ss}, from S_{sf} to S_{ns}). But during the failure detection and handling period in the state S_{sf}, should a challenge arrive, the plant would fail to initiate a trip and would go to the plant hazard state (transition r, from S_{sf} to S_{ph}) because the safety system is not able to vote for "trip" in this state.

In Figure 2, each channel can fail with its output left at either a state voting for "no trip" or a state voting for "trip", before the failure is detected and handled. When at least three channels have failed (due to either independent or common mode faults) and have left at least three votes for "no trip" (state S_{nnn}), the safety system would not respond a challenge correctly because the required 2-out-of-4 votes for "trip" never satisfy in this state. This state is regarded as the failure state of the safety system and is equivalent to state S_{sf} in Figure 1. Minimizing the occupancy probability of this state is the major goal for this model. The common mode failure rate (λ_{com}) and the failure detection and handling time (T_{dh}) are key parameters for minimizing this occupancy probability. All of the other states in this model do not affect the ability of the safety system to vote for "trip" in case a challenge arrives, and therefore none of them is designated as a failure state.

The diagram shown in Figure 3 is a rough modeling of the four components in an Eagle 21 channel. Although the four components can be further decomposed at lower levels, this further detailed modeling will not have much impact on the results because the single channel failure rate has little effect on the results, according to our sensitivity study.

3.2 Parameter Estimation

The data source was the failure reports in letter form generated for the early use of Eagle 21, including the debugging phase of its initial installation, in the Sequoyah Unit 1 and Unit 2 during a 2-year period [TVA90]. Since the initial debugging, Eagle 21 has been operating in the field without a common mode failure for at least 10 years. For the purposes of this study, this data set which reflects the reliability of Eagle 21 in the early installation and operational phase was used to estimate upper bounds of channel level failure rates.

In our previous study [Tang96], most of the failures in the above data were found to be hardware problems and only a few of them were identified to be software related. The study classified these failures by the Eagle 21 components, as shown in Table 1, which permits to estimate the λ's in Figure 3. Based on the reported dates, it was identified that the failure data represented a total of 1,130 operational unit days. Since each unit has four channels, this translates to a total of 108,480 operational channel hours. Thus, the channel component level λ's can be calculated from this information. The results are also listed in Table 1, where the mean failure rates, instead of the upper bounds of failure rates, are estimated. This is because the underlying failure data reflect the early phase reliability of Eagle 21 whose current reliability should be much better. Notice that failure rates estimated from these detected failures can be viewed as upper bounds of the Byzantine type failures modeled.

[4] An example of such failures is that the software makes a wrong judgement on an unusual combination of sensed physical parameters such that it fails to initiate a necessary trip.

[5] An example of such failures is that a problem (e.g., memory leaking) of the underlying operating system blocks the running of the application software for all channels, i.e., a common mode failure.

Table 1 Eagle 21 Component Failure Rates Estimated from Data

Eagle 21 Components	Number of Failures	Mean Failure Rate
Tester	34	3.13×10^{-4}
I/O	13	1.20×10^{-4}
Loop Processor	11	1.01×10^{-4}
Power Supply	9	8.30×10^{-5}

In addition to the above component level failure rates, an upper bound of the Eagle 21 common mode failure (simultaneous failure of at least three channels) rate can also be estimated. Since Eagle 21 has been operating in the field without a common mode failure for at least 10 years, by [Tang96], an upper bound at the 80% confidence level is given by

$$\lambda_{com} < \frac{-\ln(\alpha)}{T} = \frac{-\ln(0.2)}{10\ \mathit{years}} = 1.84\times10^{-5}/\mathit{hour}$$

For all other parameters, because of lacking real data, they were assumed to take typical or conservative values, as shown in Figures 2-4, for the demonstration purpose. Some key parameters will be varied on reasonable ranges in the following sensitivity analysis.

3.3 Sensitivity Analysis

The dependability measure to evaluate in this analysis is the plant Mean Time Between Hazards (MTBH), i.e., the mean time to state S_{ph} (Figure 1) which represents a failure of the safety system to initiate a necessary reactor trip in response to a challenge due to its computer hardware or software (design or random) faults. The MEADEP parametric analysis functionality was used on the above model to investigate the impact of the following three parameters upon the plant MTBH: (1) the safety system common mode failure rate λ_{com}, (2) the safety system failure detection and handling time T_{dh}, and (3) the probability of success upon demand P_S. When one of these parameters was selected for sensitivity study, it was varied in a reasonable range and all of the other parameters remained unchanged. Figure 4 shows the results on the parameter P_S.

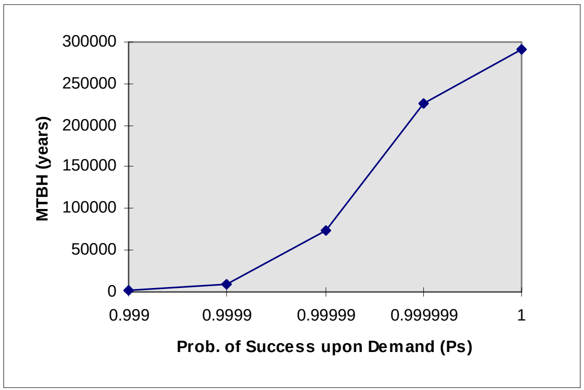

Figure 4 Sensitivity of Plant MTBH to P_S

The results showed that the plant MTBH is not very sensitive to λ_{com} and T_{dh}. For the selected parameter ranges, the variance of MTBH is about 8% and 2%. However, the plant MTBH is extremely sensitive to P_S: when P_S increases from 0.999 to 1, MTBH increases from 1000 years to 291,000 years, i.e., an increase by 290 times (Figure 4). The largest increment segment is between 0.99999 and 0.999999 (from 74,500 years to 225,600 years) and achieving a value in this range is the most rewarding. It is clear that the most important parameter is P_S, the probability of success upon demand, and achieving a high value and estimating the achieved value for this parameter should be a key effort in the system development.

One approach to estimating P_S is by means of measuring the proportion of successful test runs from test data. Stress testing techniques may be needed to accelerate the arrival rate of challenges from the plant requiring a response of the safety system. Because safety systems typically have fairly simple and well defined functions, and because these functions must generally be unambiguous and effective, their success can be described as a simple Bernoulli trial and the MEADEP DEA module can be used to determine the confidence interval for this measure. However, the validity of this approach is based on the assumption that the test environment is representative of the plant operational environment. But determining whether the test environment is sufficiently similar to the operational environment is not always straightforward.

5. Conclusion

In this paper, we discussed a measurement-based dependability modeling and evaluation tool — MEADEP. MEADEP provides a user-friendly, graphical interface on Windows 95 for non-expert users. Features of MEADEP include: converting data in various formats to the MEADEP format, graphical data presentation and parameter estimation, graphically building of dependability models, availability/ reliability calculation and parametric analysis. Use of the tool on failure data from measurements produces quantitative assessments of dependability for critical systems, while greatly reducing requirements for specialized skills in data processing, statistical analysis, and dependability modeling from the user. We also demonstrated the application of MEADEP on safety systems by modeling and analyzing a nuclear power plant's safety system based on the Eagle 21 architecture and its early field failure data. A sensitivity analysis was performed on reasonable parameter ranges. The study quantified the effect of the parameter "probability of success upon demand" on the plant MTBH and identified the most rewarding segment for this parameter.

Acknowledgments

This work was supported by the U.S. Nuclear Regulatory Commission (NRC) under Contract NRC-04-95-081. However, the opinions and viewpoints expressed herein are the authors' personal ones and do not necessarily reflect the criteria, requirements, and guidelines of the NRC.

References

[Adams84] E. N. Adams, "Optimizing Preventive Service of Software Products," *IBM Journal of Research & Development*, Jan. 1984, pp. 2-14.

[Arlat90] J. Arlat, K. Kanoun and J. C. Laprie, "Dependability Modeling and Evaluation of Software Fault Tolerant Systems," *IEEE Transactions on Computers*, Vol. 39, No. 4, April 1990, pp. 504-512.

[Butler93] R. W. Butler and G. B. Finelli, "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software," *IEEE Transactions on Software Engineering*, Vol. 19, No. 1, Jan. 1993, pp. 3-12.

[Castillo82] X. Castillo and D. P. Siewiorek, "A Workload Dependent Software Reliability Prediction Model," *Proceedings of the 12th International*

Symposium on Fault-Tolerant Computing, June 1982, pp. 279-286.

[Goswami93] K. K. Goswami and R. K. Iyer, “Simulation of Software Behavior Under Hardware Faults,” *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing*, June 1993, pp. 218-227.

[Gray90] J. Gray, “A Census of Tandem System Availability Between 1985 and 1990,” *IEEE Transactions on Reliability*, Vol. 39, No. 4, Oct. 1990, pp. 409-418.

[Hsueh88] M. C. Hsueh and R. K. Iyer, “Performability Modeling Based on Real Data: A Case Study,” *IEEE Transactions on Computers*, Vol. 37, No. 4, April 1988, pp. 478-484.

[Iyer85] R. K. Iyer and D. J. Rossetti, “Effect of System Workload on Operating System Reliability: A Study on IBM 3081,” *IEEE Transactions on Software Engineering*, Vol. 11, No. 12, Dec. 1985, pp. 1438-1448.

[Iyer96] R. K. Iyer and D. Tang, “Experimental Analysis of Computer System Dependability,” *Fault-Tolerant Computer System Design*, D. K. Pradhan (Ed.), Prentice Hall PTR, Upper Saddle River, NJ, 1996, pp. 282-392.

[Kececioglu93] D. Kececioglu, *Reliability and Life Testing Handbook*, Vol. 1 & 2, PTR Prentice Hall, Englewood Cliffs, NJ, 1993.

[Laprie84] J. C. Laprie, “Dependability Evaluation of Software Systems in Operation,” *IEEE Transactions on Software Engineering*, Vol. 10, Nov. 1984, pp. 701-714.

[Laprie85] J. C. Laprie, “Dependable Computing and Fault Tolerance: Concepts and Terminology,” *Proceedings of the 15th International Symposium on Fault-Tolerant Computing*, June 1985, pp. 2-11.

[Laprie95] J. C. Laprie, “Dependable Computing: Concepts, Limits, Challenges,” *Special Issue of the 25th International Symposium on Fault-Tolerant Computing*, June 1995, pp. 42-54.

[Lee93] I. Lee, D. Tang, R. K. Iyer and M. Hsueh, “Measurement-Based Evaluation of Operating System Fault Tolerance,” *IEEE Transactions on Reliability*, Vol. 42, No. 2, June 1993, pp. 238-249.

[Lee95] I. Lee and R. K. Iyer, “Software Dependability in the Tandem GUARDIAN System,” *IEEE Transactions on Software Engineering*, Vol. 21, No. 5, May 1995, pp. 455-467.

[Musa87] J. D. Musa, A. Iannino and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill Book Company, 1987.

[Nagle82] P. Nagle and J. A. Skrivan, *Software Reliability: Repetitive Run Experimentation and Modeling*, NASA CR-165836, Feb. 1982.

[Neil96] M. Neil, B. Littlewood and N. Fenton, “Applying Bayesian Belief Networks to System Dependability Assessment,” *Proceedings of Safety Critical Systems Club Symposium*, Springer-Verlag, Feb 1996.

[Reibman88] A. Reibman and K. S. Trivedi, “Numerical Transient Analysis of Markov Models,” *Computational Operations Research*, Vol. 15, No. 1, 1988, pp. 19-36.

[Sahner96] R. A. Sahner, K. S. Trivedi and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Experimental-Based Approach Using the SHARPE Software Package*, Kluwer Academic Publishers, 1996.

[Scott87] R. K. Scott, J. W. Gault and D. F. McAllister, “Fault-Tolerant Software Reliability Modeling,” *IEEE Transactions on Software Engineering*, Vol. 13, May 1987, pp. 582-592.

[Siewiorek92] D. P. Siewiorek and R. W. Swarz, *Reliable Computer Systems: Design and Evaluation, Digital Press*, Bedford, Mass., 1992.

[Tang95] D. Tang and M. Hecht, “Evaluation of Software Dependability Based on Stability Test Data,” *Proceedings of the 25th International Symposium on Fault-Tolerant Computing*, June 1995, pp. 434-443.

[Tang96] D. Tang, M. Hecht, H. Hecht, and R. Brill, “Measurement-Based Dependability Evaluation for Safety-Grade Digital Systems,” *Proceedings of the 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, May 1996, pp. 535-542.

[Tang97] D. Tang, J. Agron, and Y. Mendelovici, *Measurement-Based Dependability Analysis for Critical Digital Systems*, SBIR NRC-04-95-081 Phase II Final Report, SoHaR Incorporated, Sept. 1997.

[Tausworthe96] R. C. Tauworthe and M. R. Lyu, “Software Reliability Simulation,” Chapter 16 of *Handbook of Software Reliability Engineering*, M. R. Lyu, Editor, McGraw-Hill, New York, NY, 1996.

[Trivedi82] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[TVA90] TVA Letter to NRC Dated May 10, 1990, Sequoyah Nuclear Plant — *Eagle 21 Functional Upgrade Commitments*, NRC Publication Document Room, Accession Number 910715001.

[Westinghouse91] *EAGLE 21 Technical Description*, Westinghouse Electric Corporation, Process Control Division, Pittsburgh, PA, Jan. 1991.

Appendix A. Notation Used in Figures 1 - 3

S_{ns} Normal/safe state in which either both plant and safety system are functioning within technical specifications or the plant is in a safe trip (reactor is shut down safely)

S_{sp} Safety processing state in which the safety system is processing a challenge

S_{sf} Safety failure state in which the safety system is not able to respond to a challenge properly while the plant is functioning within technical specifications

S_{ph} Plant hazard state which is the result of a failure of the safety system to process a challenge successfully in terms of initiating a necessary reactor trip

P_S Probability of success upon demand, i.e., the safety system will be successful in responding a challenge (initially set to 0.9999)

r Arrival rate of challenges from the plant requiring a response of the safety system (assumed to be once a year, a typical value)

τ Challenge processing time (assumed to be a half hour, a conservative assumption)

λ_{ss} Failure rate of the safety system (evaluated from the safety system model in Figure 2)

μ_{ss} Rate for detection and handling of a safety system failure (evaluated from the safety system model in Figure 2)

μ_f Recovery rate of the plant after a hazardous event (which has no impact on the plant MTBH)

S_0 Normal state in which all the four channels are functioning properly

S_n State in which one channel has failed and the output of the failed channel votes for “no trip”

S_y State in which one channel has failed and the output of the failed channel votes for “trip”

S_{nn} State in which two channels have failed and both failed channels vote for “no trip”

S_{yn} State in which two channels have failed and one failed channel votes for “trip” and another failed channel votes for “no trip”

S_{yy} State in which two channels have failed and both failed channels vote for “trip”

S_{nnn} State in which at least three channels have failed and at least three failed channels vote for “no trip”; This state is equivalent to state S_{sf} in Figure 1 because the safety system would generate a “no trip” signal should a challenge arrive.

S_{yxx} State in which three channels have failed and at least one of the failed channels vote for “trip”

S_{trip} Plant trip state (reactor is shut down)

P_n Probability that the channel output votes for “no trip”, given a channel failure (assumed to be 0.5)

λ_c Failure rate of a channel (evaluated from the channel model in Figure 3)

μ_c Recovery rate of a channel (evaluated from the channel model in Figure 3)

λ_{com} Common mode failure rate for the safety system (80% confidence upper bound based on no common mode failure for 10 years)

T_{dh} Failure detection and handling time, given that at least three channels have failed (assumed to be one hour)

T_{trip} Plant trip duration (assumed to be 50 hours)

IO The I/O component of a channel

LP The Loop Processor component of a channel

Tester The Tester component of a channel

Power The Power supply component of a channel

λ, μ Failure rate and recovery rate for the above components (λ is estimated from failure data and $1/\mu$ is assumed to be one hour)