Proceedings of the 12th Annual Conference on Computer Assurance, June 16-20, 1997, Gaitherburg, Maryland, USA

Quantitative Reliability and Availability Assessment for Critical Systems Including Software

Myron Hecht, Dong Tang, and Herbert Hecht
SoHaR Incorporated, Beverly Hills, CA
and
Robert W. Brill[1]
U.S. Nuclear Regulatory Commission, Washington, DC

Abstract

In many cases, it is possible to derive a quantitative reliability or availability assessment for systems containing software with the appropriate use of system-level measurement-based modeling and supporting data. This paper demonstrates the system-level measurement based approach using a simplified safety protection system example. The approach is contrasted with other software reliability prediction methodologies. The treatment of multiple correlated and common mode failures, systematic failures, and degraded states are also discussed. Finally a tool called MEADEP, which is now under development, is described. The objective of the tool is to reduce the system-level measurement-based approach to a practical task that can be performed on systems with element failure rates as low as 10^{-6} per hour.

1. Introduction

When safety systems and control systems with safety significance were initially developed, analog electronics was the dominant technology. With this technology, there were accepted approaches and techniques to calculate quantitative system reliability. An examples is the Joint International Civil Aviation Organization and Federal Aviation Regulation (JAR/FAR) 25.1309. Although digital (i.e., software-based) systems are replacing analog instrumentation and control systems, the requirements remain in force. Moreover, newer industry standards explicitly applicable to software-based safety systems also incorporate quantitative requirements. An examples is Sec. 7.5.2.6 of Draft IEC 1508 -1, "Functional safety: safety related systems" (November, 1995) which addresses programmable electronic systems used in general safety applications. Perhaps even more important than government regulations or industry standards are user requirements for newer systems and public expectations for progressively higher levels of dependability. Thus, there is a need for methodologies that can arrive at a defensible quantitative estimate of integrated hardware/ software reliability and availability. These methods are necessary to properly assess the acceptability of digital systems in critical applications and demonstrate compliance with regulations, standards, and expectations.

In many cases, it is possible to use a combination of measurement and reliability/availability models to develop a quantitative assessment. Previous research on mature fault tolerant and other high integrity systems has shown that residual software faults lead to a failure behavior which can be characterized by an MTBF [Nagle82, Adams84, Hsueh88] and that a majority of such failures could be recovered from by the use of physical redundancy [Gray90, Lee95, Tang95]. These results provide a basis for modeling software failures as a stochastic process in systems where (1) the requirements definition (including specification of off-normal states and events in the operating environment) is substantially correct and complete, (2) a disciplined development, has been effective in eliminating defects with a high degree of determinism and repeatability, and (3) testing and past operational history are largely representative of future performance.

[1] This research was supported in part by the U.S. Nuclear Regulatory Commission (NRC) under contract NRC 04-95-081. The opinions expressed by the author are his personal ones and do not necessarily reflect the official position of the NRC.

This paper describes a methodology to perform quantitative reliability assessment in systems where the predominant failure mechanism is related to stochastic failures. Because both data and models are used, we call this approach a *measurement-based system level* method. Section 2 contrasts other software reliability prediction methods with the measurement-based system-level approach. In section 3, an example using a simplified safety protection system demonstrates how a measurement-based system-level approach can be used to provide a quantitative estimate of availability. Section 4 discusses the issues raised and addressed by the example. Section 5 describes a tool which supports the approach, and section 6 provides concluding remarks.

2. Approaches for Assessing Software Reliability

There are many frameworks and taxonomies for software reliability models [Farr96]. For the purposes of this paper, we group these approaches into four major categories: ignoring software failures entirely, fault density models, reliability growth models, and system-level measurement-based modeling. Table 1 provides a summary of these categories.

Table 1. Comparison of Alternative Approaches for Assessing Software Reliability in a Critical System

Method category	Measures software failure rates	Incorporates system architecture	Accounts for common cause failures
Ignoring Software failures	no	yes	no
Fault density	not directly	no	no
Reliability growth	yes	not directly	not directly
System level measurement-based	yes	yes	yes

The column headings used for their characterization are as follows:

- *Uses software failure rates*: Whether some measure of software failure rates or intervals between failures (per execution, processor time, or external time) is incorporated into calculations used by the approach. Techniques which incorporate actual measures of software failure rates are preferable to those which infer it through project characteristics or other indirect measures because of greater accuracy.

- *Incorporates system architecture:* Whether the method accounts for redundancy, component organizations, system reconfigurations, and other aspects of system architecture. Techniques which account for system architecture are preferable because they more closely represent the actual system.

- *Accounts for common cause or multiple correlated failures*: A major concern of software is the extent to which a single failure -- or a set of multiple correlated failures can disable any redundancy present in a system. In fault tolerant systems, the term fault detection and recovery coverage, or simply *coverage* has been defined to address this issue. Coverage is the conditional probability that a system will recover given that a hardware or software failure has occurred in a recoverable element[2]. Coverage has a very dramatic effect on system-level availability and reliability [Trivedi82]. Methods which account for common cause or correlated failures are preferable to those which do not.

The following subsections provide the rationale for Table 1.

2.1 Ignoring software failures

For the reasons described below, fault density and reliability growth modeling have been viewed with skepticism [Brocklehurst96]. Thus, many standards and consensus guidelines which deal with software based critical systems have avoided quantitatively addressing software failures. For example, RTCA DO 178B [RTCA92], the standard used for certifying digital avionics, posits that specific development practices (e.g., branch coverage testing or levels of configuration management) will result in attaining certain levels of safety assurance. Similarly, the UK Health and Safety Executive (HSE) Guidelines on Programmable Electronic Systems in Safety Related Applications [UKHSE87] distinguishes "random hardware failures" from "systematic errors" which include errors or omissions in software. Such approaches are not satisfactory because they explicitly avoid any quantitative statement on system reliability.

2.2 Software Fault Density Models

Fault density models attempt to predict the number of

[2] Fault detection and recovery coverage can be measured during testing using fault injection, and during operation, by observing the number of successful recoveries versus total recovery actions.

faults per line of code (or thousand lines of code) based on readily measurable characteristics of the code and the project. The U.S. Air Force Rome Laboratory sponsored research into developing predictions of fault density (i.e., number of coding defects per thousand lines of source code) which they could then transform into reliability measures, such as failure rates [Friedman92]. The predictions of fault density are based on the characteristics of the application, development environment, extent of reuse and other factors. This study and other sources contain data on expected fault density which currently ranges from 1 to 5 faults per thousand source lines of code (KSLOC). The assumption behind fault density-based prediction models is that as the number of software coding defects (faults) increases, reliability decreases. The translation of fault density to failure rate requires assumptions about the probability of encountering a fault during execution. This probability can vary widely, depending on the location and nature of the fault. The empirical data on this probability that are currently available do not support very accurate predictions of the failure rate. For instance, a study of failure data from the stability tests of an air traffic control software system showed that failure rates attributed to different faults can differ by two orders of magnitude [Tang95]. Thus, such methods can not be viewed as being adequate for safety critical systems.

2.3 Software Reliability Growth Models

Reliability growth models have been an active area of research since the early 1970s [Farr93]. Examples are the Schneidewind model, the generalized exponential model, the Musa/Okumoto Logarithmic Poisson model, and the Littlewood/Verrall model [ANSI92]. These models use measured trends of failure rates (or change in intervals between failures) and extrapolate them to future operation. In most cases, they evaluate the reduction in failure frequency during successive developmental test intervals to estimate the number of defects or software reliability at the conclusion of the test (and sometimes into operational deployment). It is expected that overall, the hazard rate will decrease over time, but that there are discontinuities as each failure occurs. However, as the program runs for more time, there is increasing confidence in the reliability of the program.

Software reliability growth models can be used as part of developmental testing in order to determine whether non-critical software is ready for release [Wood96]. However, for the purposes of high criticality systems, this class of software reliability models have the following drawbacks:

- *Difficulty in accumulating the data required to demonstrate low failure rates.* Applications of these models have all been demonstrated using real data from software with typical failure rates of 10^{-1} to 10^{-3} per hour [Abdel-Ghaly86, Musa87]. While this failure rate may be acceptable for some types of applications, the failure rate for safety critical applications must be much lower. However, for life critical systems, it would take 10^{8} to 10^{10} hours (thousands of years) of testing to demonstrate a failure rate of 10^{-7} to 10^{-9} per hour assuming one copy of software would be tested and one failure would be observed [Butler93]. Even if 10 copies of the software are tested concurrently, it would still take hundreds of years. A related issue is the issue of the operational profile. In many types of safety systems, the most important reliability-related figure of merit is the probably of success upon demand. This is the probability that the system mitigates or controls the consequences of a process or system transient *when the transient occurs*, not the amount of time that the system runs without a failure under normal operating conditions. In such safety systems, a large amount of the functionality, i.e., recovering and managing a transient event makes up a very small portion of the total operating time. Safety system testing and reliability estimation must account for this in a different way than the time-based data emerging from these models.

- *No accounting for partitioning, redundancy and fault tolerant system architectures:* A characteristic of nearly all safety-related system is the presence of physical redundancy[3], failure containment regions, logical redundancy, a backup analog system, and other techniques. The reliability growth-based models regard the software as a monolithic black box and does not account for redundancy and a finite probability that an automatic recovery action within the system will be successful in allowing continued execution.

[3] Several studies [Lee93, Tang95] have shown that 80 to 95 percent of software failures in real-time fault-tolerant systems are recoverable by redundant processes.

2.4 System-Level Measurement Based Modeling

System-level measurement based modeling provides a quantitative estimate of reliability and availability using a combination of (1) system-level reliability/ availability models (primarily reliability block and Markov models) and (2) representative data to estimate the values of parameters within the model.

Measurement based modeling methods for integrated hardware and software systems developed over the past 15 years [Iyer93] address many of the drawbacks identified above. A systems view (rather than a strictly software view) of reliability and availability is an important element in the method. The integrated application software tasks, operating system kernels or executives, and hardware components all are elements affecting operational dependability. System-level reliability and availability are then estimated by models of the system structure, using measurement-based parameters for each component [Tang95]. This is a perspective similar to that taken by classical reliability block diagram (RBD) modeling. However, unlike RBDs, the measurement based modeling methodology that will be described below does not assume total independence of elements, nor does it assume that recovery will always occur. Operational data are used to determine correlated failure and recovery probabilities, as well as failure rates and recovery times. As noted in the introduction, the validity of the approach is determined by how well the system conforms to the following assumptions:

1. The system has been developed using a disciplined development process which ensures traceability and removal of reproducible failures found during verification, validation, and testing.

2. Requirements are well understood by the designers, programmers, and testers.

3. Previous operational data and test data are representative of the actual operating environment, and sufficient testing time is available to allow for meeting the test coverage criteria.

The basis for the validity of stochastic models under these assumptions was also discussed in the introduction. Section 4.7 contains additional discussion about the limitations of this approach when the above assumptions are less applicable.

3. Application of the System-Level Measurement-Based Approach

Arriving at a quantitative reliability or availability estimate requires a (1) model and (2) data to substantiate the values of the parameters used in the model. The following subsections describe the approach for a simplified digital (i.e. microprocessor and software-based) safety protection system (SPS) as an example. An SPS monitors sensors in a plant, and trips circuit breakers if sensor values fall outside of an allowable range. The action of the circuit breaker being tripped disables the plant control system and stops the process. Figure 1 shows a simplified SPS diagram used in this analysis.

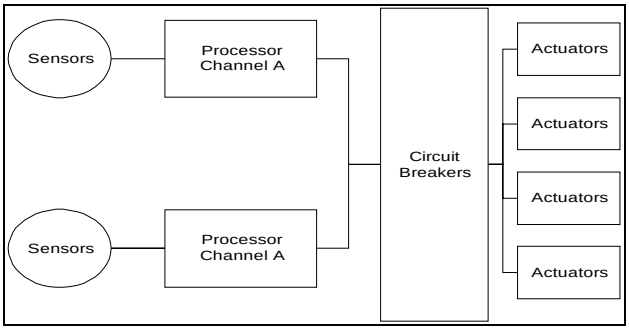

Figure 1 Simplified SPS

3.1 Description of the SPS model

Figure 2 shows the top level Markov model for the SPS. The model has two major components: the plant and the SPS itself.

The notation used in the model is as follows:

N Normal state in which both safety and plant are functioning within technical specifications

SP Safety processing state in which the SPS is responding to sensor values outside of the allowable range (such an event is called a *challenge*)

SF Safety failure state in which the SPS is not available for responding to a challenge while the plant is functioning within technical specifications

F System failure state in which the SPS has failed to process the challenge

P_S Probability of success upon demand, i.e., that the SPS will be successful in responding to a challenge

R Arrival rate of challenges from the plant requiring a response of the SPS

τ Challenge processing time

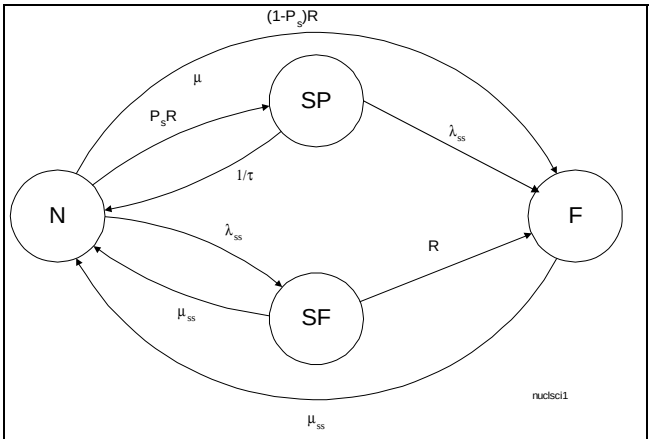

Figure 2. SPS Top Level Markov Model

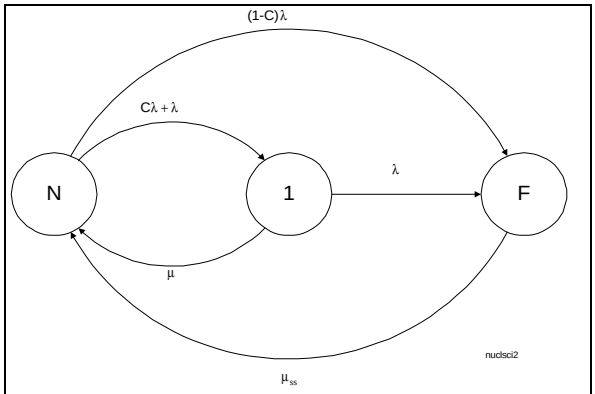

Figure 3 A Markov Model for a System with a Standby Channel

λ_{ss} Failure rate of the SPS system (evaluated from a submodel shown in Fig. 3)

μ_{ss} Recovery rate of the SPS after a failure

In the normal state, if a challenge arrives, the SPS will respond to it successfully with probability P_S and go to the safety processing state SP (modeled by the transition $P_S R$, from N to SP). During the safety processing, if the SPS fails due to its hardware or software problems, the system will fail (transition λ_{SS}, from SP to F). Otherwise, the SPS will go back to the normal state after the mean processing time (transition $1/\tau$, from SP to N). When a challenge arrives in the normal state, the SPS may respond to it unsuccessfully and go to the system failure state (transition $(1-P_S)R$, from N to F).

Sometimes the SPS fails in the normal state and goes into the safety failure state SF (transition λ_{SS}, from N to SF). The SPS will go back to the normal state with rate μ_{SS} (transition μ_{SS}, from SF to N). But during the recovery period in the state SF, if a challenge arrives, the system will fail (transition R, from SF to F) because the SPS is not available for use in this state.

The parameter λ_{SS} is the failure rate of the SPS hardware and software due to either multiple simultaneous random failures or due to a single common cause event. Estimation of this parameter is incorporated into a lower level model, or submodel shown in Figure 3[4]. The SPS is assumed to consist of two channels: a primary (or "hot") channel, and a standby channel.

[4] The circuit breakers are not included to reduce the complexity of the model. They would be included as series elements with the submodel in Figure 3.

The notation used in the model is as follows.

N	Normal state in which both primary and standby channels are functioning
1	State in which one channel has failed and another channel is functioning
F	Failure state in which both primary and standby channels have failed. This state is equivalent to SF in Fig. 2; the total transition rate from N to F represents λ_{SS} in Fig. 2.
λ	Failure rate of a channel
μ	Recovery rate of a channel
μ_{SS}	Recovery rate of the SPS after a failure
C	Probability that switchover to the standby will be successful when the primary channel fails. The complement of this quantity, or 1-C, is the probability of a common cause or correlated failure at the system level that defeats both channels.

In the normal state, either the primary or the standby channel can fail. If the primary channel fails, the system will switchover to the standby successfully with a probability C (coverage). This is modeled by a transition from state N to state 1 with rate $C\lambda$. If the standby channel fails, the system will also go to state 1, but with a transition rate because no switchover will be involved. Thus, the aggregate rate of the transition from state N to state 1 is $C\lambda + \lambda$. Note that this rate represents single channel failures and does not indicate any failure of multiple channels. If C=1, the aggregate rate would be 2λ, which implies failure independence (i.e., single channel failures arrive at rate 2λ and will not cause a second failure).

During the recovery of the failed channel (state 1), if another channel also fails (independently), the system will fail, which is modeled by the transition from state 1 to state F. There is a possibility, 1-C, that the switchover to the standby will not be successful, which leads the system directly to the failure state F. This is modeled by the transition from state N to state F with rate (1-C) . After a system failure, the system will recover from the failure at a rate μ_{SS}, modeled by the transition from state F to state N marked by μ_{SS}.

Other models would be appropriate for different redundancy schemes. For example, in a four-channel reactor trip system, a 2-out-of-4 reliability block diagram model could be used to handle multiple independent randomly occurring failures. Common cause and correlated failures could be modeled by Markov models similar to that in Figure 3. The output of these models could then be incorporated into the higher level Markov model shown in Figure 2.

Deterministic failures in the logic of the software are accounted for by P_s, the probability of success upon demand. P_s, can be estimated by measuring the proportion of successful test runs from test data and using the binomial distribution to determine the confidence interval. Because SPSs 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.

3.2 Estimation of the parameters

Estimation for the actual parameters in the model can be performed using established methods in standard reliability analysis. For example, if failure arrivals can be assumed to follow an exponential distribution, the upper limit (λ_U) and the lower limit (λ_L) of failure rate can be determined by the following relations [Kececioglu93]:

$$\lambda_U = \chi^2_{1-\alpha;\, 2n+2} /2T$$
$$\lambda_L = \chi^2_{\alpha;\, 2n} /2T$$

where n is the number of observed failures, T is the test time, α is the level of significance, and χ^2 represents the chi-squared distribution. If no failures were observed during a test, then the upper limit of the failure rate is [Tang95]

$$\lambda_U = -ln(\alpha)/T$$

Estimates of the confidence limits of C and P_s can be determined by binomial distributions. If the number of successes, s, in n trials is greater than zero and less than n, the upper limit (P_U) and lower limit (P_L) of P can be approximated by [Kececioglu93]

$$P_U = 1/(1+((n-s)/(s+1))F_{\alpha;\, 2(n-s);\, 2s+2})$$
$$P_L = 1/(1+((n-s+1)/s)F_{1-\alpha;\, 2(n-s)+2;\, 2s})$$

where F represents the F distribution and α is the significance level. If s equals n, a conservative lower limit is given by [Tang95]

$$P_L = \alpha^{1/n}$$

where α is the significance level. Further details on the data analysis process can be found in earlier research that the authors have performed [Tang95b, Tang96].

4. Discussion

This section discusses some of the issues which are addressed by the SPS model described in the previous section, the validity and use of failure data, and past experience with the system-level measurement-based evaluation approach on several air traffic control and nuclear reactor safety/control systems.

4.1 Intermittent vs. Continuous operation

The SPS described in the previous section has components which operate intermittently (the functions which actuate the trip) and other functions which operate continuously. The dual modes of operation were addressed in Figures 2 and 3. Intermittent operation was modeled by the transition between State N (the normal operation state) and state SF (safety function state) in Figure 2. The reliability-related figure of merit for this mode of operation is the probability of success upon demand, represented by the term P_S in Figure 2. Continuous operation is represented in Figure 3. For such a system, availability is the most significant figure of merit, and λ and C are key parameters.

Continuously operating digital control systems, whether used for controlling vehicles or industrial processes should be modeled differently. Rather than having one safety processing state (SP) as shown in Figure 2, there may be a large number of such states, and they may all be running continuously. Essential functions which are continuously running (e.g., an operating system) can be combined into the N and 1 states of Figure 3, with the failure rate adjusted appropriately.

4.2 Correlated and Common Cause Failures

Many types of correlated common cause failures are addressed by the model shown in Figures 2 and 3. In Figure 2, these failures are accounted for by the N to F transitions. The transitions describe an event in which a functional system experiences a failure (or set of failures) which results in the loss of the entire system. In this model, the disabling of the safety system does not necessarily result in an unsafe condition because there may not be a challenge requiring the intervention of the safety system during the time that the system has failed[5]. The SF to F transition, however, is the joint event that the SPS is down AND the system experience a transient or other challenge requiring the intervention of the safety system.

At the lower level of the model (Figure 3), the coverage factor C[6] accounts for common cause failures. The upper and lower limits of coverage can be calculated using methods of proportions. The extent of correlated failures (if any) can also be determined using correlation matrices. However, in the absence of any failures, calculation of availability using the lower limit of coverage may result in an unacceptably low reliability or availability. In most of the systems we have analyzed thus far using this approach, there has been sufficient data to empirically determine the probability of a correlated failure (i.e., the conditional probability of a failed recovery action given a failure occurred). However, this is not always the case. When the correlation information is unknown after a period of measurement, we have used values from analyses of comparable systems as an estimate. However, more research is necessary to develop unambiguous approaches for identifying equivalent systems or other means of deriving reasonably conservative values for coverage.

In many cases, multiple failures which cause catastrophic system effects (i.e., totally disable the system), have *not* yet been observed, and a different approach is necessary. Earlier work has shown that catastrophic failures in well-tested systems (not only digital systems) usually result from the coincidence of a number of independent conditions that are individually tolerable or, at least, non-catastrophic [Hecht94]. Some examples are documented in [Hecht95]. The rate of occurrence of non-catastrophic predecessors can be often estimated based on reported failures. An estimate of the probability of a catastrophic failure can then be determined by the combined probability of two or more individual predecessor events. Whether the combined probability is a simple product of two individual probabilities or another function of both individual probabilities and correlation parameters can be determined by a correlation analysis.

4.3 Validity of the data used for measurement-based evaluation

Obtaining adequate data from which to assess reliability and availability is critical to any measurement-based methodology. This obvious principle can be difficult to implement in practice for dependability assessments because of the constraints of an expensive testing program or impending project deadlines. Adequate data means monitoring and recording events of interest such as failures and recoveries of components, as well as performance parameters of the target system while it is operating under representative workloads. It also means collecting data on failure modes so that an assessment of the importance of deterministic failures can be made. The events and parameters to be collected should be representative of the system operation and meaningful for the assessment of the system. Measurements should be made for a sufficient period to yield statistically significant results[7]. Operating logs should include information about the location, time and type of the error, the system state at the time of failure or abnormal operation, and error recovery (e.g., retry) information where applicable.

4.4 Uses of system failure data

For the measurement-based methodology, the primary use of the failure data is for the determination of the model parameters defined in section 3. However, the data can also be used to (1) evaluate the validity of the underlying assumptions of the measurement-based approach for the specific system under analysis and (2) gain additional insight into the failure behavior of the system under analysis. Specific additional ways in which the failure data can be used are:

Analysis of anticipated vs. observed failure modes and severities: As part of the early operational evaluation of a safety system, it is necessary to compare observed failure modes and severities with those anticipated in the V&V and testing activities. The extent of discrepancies between anticipated and observed failure behavior determine the validity of the results of this prediction -- as well as of the

[5] In other system designs where active control is continuously necessary (e.g., an aircraft or spacecraft), this model would not be appropriate because loss of the system would in and of itself be an unsafe state.

[6] Coverage as defined in this paper represents the conditional probability that a failure will be detected and recovered given that the failure occurred.

[7] That is, enough data so that at the specified confidence limits, the parameter values in the model will yield results of sufficient precision that a determination of reliability or availability suitability an be made.

accompanying system analysis and design activities that occurred during development.

Testing for the presence of systematic errors. Systematic failures, i.e., failures traceable to an incorrect design or a requirements statement -- are not addressed by this approach. A qualitative analysis is necessary to establish the relative importance of such systematic errors. Evidence of a significant proportion of such failures would limit the usefulness and validity of any stochastic based approach.

Determining whether workload effects are present. If workload data are available, the presence of any correlation of failure rates with workload should be undertaken. Past analyses of workload and failure data collected from IBM mainframes [Butner80] and DEC minicomputers [Castillo81] revealed that the average system failure rate is strongly correlated with the average workload on the system. However, for a safety system, there should be no such correlation because the load should be small relative to the capacity of the CPU, I/O, and RAM[8].

Checking for the presence of correlated failures. This analysis is to check whether there is evidence of correlated failures among processors. Recent studies of data from DEC [Tang92] and Tandem [Lee93] systems showed that correlated failures across processors are significant in multicomputers, and their impact on dependability is significant.

4.5 Accounting for Degraded Modes of Operation

In the SPS example described above, the system was either functioning or failed. However, in other systems, a failure may result in a degradation, but not necessarily a total loss of the function. This can be handled through the concept of a *reward* function associated with the Markov process. Figure 4 shows an example of a portion of a model for an air traffic control system. In this model, there are 12 radars which provide surveillance coverage for a certain airspace. If all 12 radars are operational (state S0 in Figure 4), there is 100% airspace surveillance coverage. As the number of failed radars increases, the airspace surveillance coverage, i.e., *reward*, decreases. Thus, in the figure, the reward for a single site failure (R1) is approximately 99.3%, for two simultaneous failures, it is 98.0%, etc. The expected level of airspace surveillance coverage is the availability evaluated from this Markov reward model.

[8] Under the assumptions described in section 2, the requirements are assumed to be well understood.

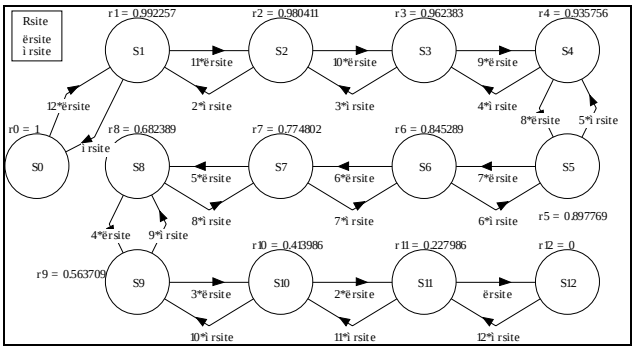

Figure 4. Markov Model of a set of radars demonstrating degraded performance

The approach can also be used for systems where there are multiple functions which can fail separately without causing a system level failure, or for multiprocessing systems where failures result in a loss of capacity but not a total loss of functionality.

4.6 Other Applications of the Measurement-Based Approach

Air Traffic Control. Air traffic control (ATC) applications differ qualitatively from the safety protection system discussed in section 3. ATC systems are complex whereas the SPS is simple; ATC systems are operational and active in all modes continuously whereas the SPS performs the majority of its function intermittently. These differences cause a changes in the models which are used, but not the principal of measurement-based analysis. The methodology has been used to evaluate availability for the current generation of en-route air traffic control [SoHaR96] systems. In this study, outages from the first 10 months of 1995 were analyzed for computers, radars, communication hardware, and displays. A hierarchical model consisting of 15 submodels was developed to represent the system. The result of the analysis was an airspace coverage availability of 0.99991 (i.e., the probability that a given section of airspace would have complete radar, communication, and computer services). In the earlier study on an air traffic control system in the late developmental testing phase [Tang95], the automated data collection capability of the test system was the source of the data. These characteristics permitted estimation of the coverage of the fault tolerance provisions and evaluation of system availability based on a model similar to that shown in Figure 3. The measurement-based approach has also been applied to the FAA Voice Switching and Control System (VSCS) installed in 12 major U.S. air traffic control centers. The result of the system availability assessment performed in March of 1996 was 0.99999, or a downtime of 32 seconds per year. The downtimes were dominated by causes related to software

failures. When combined with the hardware-only backup, the downtime was reduced to 1.6 seconds per year.

Nuclear Reactor Protection System. The methodology has been applied to a nuclear reactor plant protection system (Eagle 21) during early operation [Tang95b]. In the plant protection system, data were collected during the first operational period of Westinghouse Eagle 21 systems at the Sequoyah Nuclear Plant [TVA90]. The system was operational and therefore there was sufficient operating time to determine lower confidence limits on failure rates. Because operating time measurements were available, it was possible to estimate an upper bound for failure rates. However, the data which could be used to determine P_S were sparse because of the lack of safety events (challenges) in the operational environment. Thus, additional test data would be needed in order to provide a sufficiently large sample from which this parameter could be calculated.

4.7 Limitations of System-Level Measurement-Based Modeling

The assumptions under which the system-level measurement based approach is applicable were identified in section 2.4. The basic premise in these assumptions is that in a well tested system with a mature development process, the predominant failure mechanisms are due to the interaction of randomly arriving inputs (in a stable operating environment) interacting with residual defects in the code or the hardware. This premise is reasonable because randomly occurring difficult to reproduce failures (sometimes referred to as "Heisenbugs" after the German physicist responsible for the uncertainty principle) are much more difficult to resolve during development. The premise should be validated for each system under analysis by examining the causes of failures as suggested in section 4.4. The measurement-based approach will be less effective for software in high integrity applications where requirements are not completely understood by the developers or where the testing program does not accurately represent the intended application.

A comparison between the strengths of the measurement-based assessment and the use of disciplined developmental practices reveals how the two approaches complement each other. Requirements deficiencies, design errors, and deterministic failures due to coding errors are well suited to the testing, verification and validation processes in disciplined development environments. Infrequently occurring irreproducible failures are more difficult to control in the development process. On the other hand, they are much more amenable to the measurement-based approach defined here. The frequency of failures related to these residual faults (in either hardware or software) then become the determinants of reliability and availability.

The current practice of measurement-based evaluation approach is still limited to the failure rate of 10^{-2} to 10^{-6} per hour and the availability of three to five 9's (0.999 to 0.99999). This limitation can be shown by the evaluation for the VSCS system. If no major failures occurs, it would take 15 years of normal operation for 21 VSCS systems to demonstrate an availability of the required seven 9's at the 80% confidence level. Thus, to assess extra high reliability and availability, accelerated testing and assessment methods may be required, in addition to the measurement-based evaluation approach.

5. The MEADEP Software Tool

A PC-based modeling tool called MEADEP (MEAsure DEPendability), is being developed to implement the system-level measurement-based methodology. The overall design objectives of the tool are:

- *repeatability:* create a tool from which repeatable reliability and availability analyses can be performed,
- *accessibility:* facilitate the use of system level measurement-based method by non-experts, and
- *affordability:* reduce the cost of such analyses so that they become an integral part of all systems where reliability or availability are important considerations.

In the first phase of the development, the feasibility of developing a tool to perform measurement-based system modeling was demonstrated using the Eagle 21 reactor protection system and several of the air traffic control applications described above [Tang95, Tang95b]. Additional development in the second phase has resulted in a more mature, robust, and capable tool prototype[9].

Currently, MEADEP includes a data editing and processing module, a statistical analysis module, a model generation module, and a model solution module. The following capabilities and features to support reliability and availability analysis have been implemented.

- *Model libraries:* A library of files called *model profiles* has been developed and will be included in the final product. Model profiles define the structure of a reliability/availability model for a particular system, but

[9] Scheduled completion is August, 1997

do not contain the parameter values are included. The library will reduce the effort needed to develop system-level models. The library is expandable to accommodate all likely dependability models of interest to customers.:

- *Graphical input of reliability models:* A graphical editor with "drag and drop" capabilities has been developed specifically for MEADEP. With this capability, users will be able to easily able to create and edit models of reliability block diagrams, Markov chains, and k-out-of-n blocks. The "drag and drop" interface takes only a few minutes to learn. Figure 4 (the radar coverage Markov reward model) was generated with MEADEP.
- *Hierarchical modeling:* MEADEP allows a complex model to be decomposed in hierarchical elements. With hierarchical modeling, users will be able to specify the system in progressively lower levels of detail. The benefits are (a) reducing the labor and likelihood of errors large models, and (b) allowing lower level "building block" models to be reused in higher level designs. An simple example of hierarchical modeling was discussed in section 3.
- *Estimation of model parameters directly from failure data*: MEADEP uses statistical routines to analyze failure data and determine the values of parameters. In other words, a user will be able to initiate a calculation of results after defining a model and creating a project failure data base. The integration of statistical and reliability/availability analysis capabilities into a single tool is an important benefit in making sophisticated capabilities available at low cost. It allow facilitates periodic analyses being performed using a the set of failure data for each period.
- *Support for sensitivity analyses:* MEADEP allows a model to be run with a range of user-specified values and user-specified parameters. This capability is important to allow the user to evaluate the overall system availability over an entire confidence interval, to determine the bottlenecks in reliability or availability, or to demonstrate the impact of parameters over which there is uncertainty.
- *Multiple graphical presentations of data*: MEADEP provides a number of different graphical formats to allow the user to display the relative contributions of system components or sites to overall availability, show the distribution of mean time between failures (MTBF) to assess the degree to which the failure data follow the exponential distribution (an assumption used by Markov models). Numerical results from statistical analysis and model solution modules can also be easily "cut and pasted" into spreadsheet processors.
- *Flexible data input*: The data entry module can accept structured data in any format accessible by ODBC (including delimited text files). The user can associate any field name in the input data to one of the data items required for the analysis. Figure 5 shows an example of such an association in which the field called "ACN200 problem number", which contains date information, is associated with the "Date" field of the MEADEP format. Furthermore, data items can be substituted and aggregated. For example, if some users have specified the same system using different names, the user interface allows the interactive identification and changing of such field names.

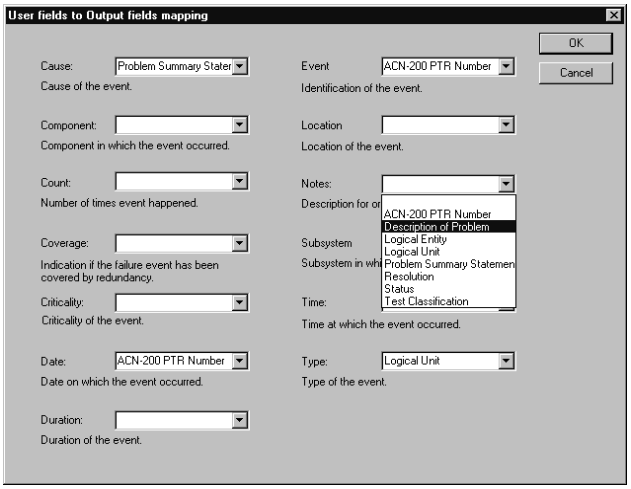

Figure 5 Data entry screen from MEADEP for failure data containing items required by an air traffic control system model

6. Conclusion

The system-level measurement-based methodology described in this paper consists of feasible methods in data collection and processing, statistical analysis, and reliability/availability modeling. It assesses system reliability/availability based on data from measurements with a specified confidence level. System-level reliability/availability models can also identify the elements where reliability improvement will provide the greatest benefit.

The validity of the quantitative assessment derived through the measurement-based system level approach is determined by the extent to which the system failure behavior conforms to a stochastic process[10]. As was discussed earlier, evidence from large studies of other high availability computer systems provides substantial evidence of such behavior. However, if a qualitative analysis of failures shows that the majority are due to inadequate requirements (reflecting a lack of understanding of the operational profile), maintenance or calibration errors, or environmental problems, then the validity of the assessment will be questionable. While the measurement-based system level method does account for some degree of incorrect responses and common mode failures, it does not adequately describe failure behavior when such failures are dominant Additional study of reliability prediction in the presence of a large component of systematic failures is necessary in order to completely address this issue.

The current measurement-based evaluation approach is limited to the systems with failure rates higher than 10^{-6} per hour and the availability below 0.999999. To assess extra high reliability/availability, accelerated testing and assessment methods may be required, in addition to the measurement-based evaluation approach discussed in this paper.

Acknowledgments

Portions of this work were supported by Nuclear Regulatory Commission contract NRC 04-94-038 and by the Federal Aviation Administration through contract DTFA01-93-Y-0169, subcontract 96-107. The authors wish to express their appreciation for the support, encouragement, and comments of Ms. Loni Czekalski, Dr. Jady Handal, and Mr. John Williams of the Federal Aviation Administration.

References

[Abdel-Ghaly86] A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, "Evaluation of Competing Software Reliability Predictions", *IEEE Transactions on Software Engineering*, vol SE-12 no. 9, Sept. 1986, pp. 950-967.

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

[ANSI92] *American National Standard, Recommended Practice for Software Reliability*, American National Standards Institute, ANSI/AIAA R-013-1992.

[Brocklehurst96] Sarah Brocklehurst and Bev Littlewood, "Techniques for Prediction Analysis and Recalibration", in *Handbook of Software Reliability Engineering*, M. Lyu, Editor, McGraw Hill, New York, 1996.

[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 SE19 no. 1, January 1993, pp. 3 - 12.

[Butner80] S.E. Butner and R.K. Iyer, "A Statistical Study of Reliability and System Load at SLAC," *Proc. 10th Int. Symp. Fault-Tolerant Computing*, Oct. 1980, pp. 207-209.

[Castillo81], X. Castillo and D.P. Siewiorek, "Workload, Performance, and Reliability of Digital Computer Systems," *Proc. 11th Int. Symp. Fault-Tolerant Computing*, July 1981, pp. 84-89.

[DoD87] U.S. Department of Defense, *Reliability Test Methods, Plans, and Environments for Engineering Development, Qualification, and Production*, MIL HDBK-781, 14 July, 1987.

[Farr96] W. Farr, "Statistical Reliability Modeling Survey," *Handbook of Software Reliability Engineering*, M. Lyu, Editor, McGraw-Hill, New York, pp. 71-117, 1996.

[Friedman92] Michael Friedman, *Methodology for Software Reliability Prediction and Assessment*, Report RL-TR-92-52, Rome Laboratory 1992 (2 volumes).

[Gray90] J. Gray, "A Census of Tandem System Availability Between 1985 and 1990", *IEEE Transactions on Reliability*, May 1990, pp. 409-418.

[Hecht94] H. Hecht and P. Crane, "Rare Conditions and their Effect on Software Failures," *Proceedings of the 1994 Reliability and Maintainability Symposium*, January 1994, pp. 334 - 337.

[Hecht95] H. Hecht, M. Hecht, G. Dinsmore, et. al., *Verification and Validation Guidelines for High Integrity Systems*, U.S. Nuclear Regulatory Commission and Electric Power Research Institute Report NUREG/CR-6293, March, 1996.

[Hecht96] H. Hecht and M. Hecht, "Qualitative Interpretation of Software Test Data," *Proc. International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability*, Beijing, China, July 1996, pp. 175-181.

[Hsueh88] M.C. Hsueh and R Iyer, "Performability modeling based on real data: A case study", *IEEE Transactions on Computers*, vol. 37 no. 4, April 1988, pp. 478-484.

[IEEE91] *Standard Criteria for Safety Systems for Nuclear Power Generating Stations*, IEEE Std 603-1991, available from IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855.

[IEEE93] *Standard for Digital Computers in Safety Systems of Nuclear Power Generating Stations*, IEEE Std 7-4.3.2-1993,

[10] these same assumptions and limitations are applicable to hardware reliability predictions.

available from IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855.

[Iyer93] 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.

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

[Lee93] I. Lee, D. Tang, R.K. Iyer, and M.C. Hsueh, "Measurement-Based Evaluation of Operating System Fault Tolerance," *IEEE Transactions on Reliability*, June 1993, pp. 238-249.

[Lee95] Inwhan Lee and Ravi K. Iyer, "Software Dependability in the Tandem GUARDIAN System," in IEEE Transactions on Software Engineering, May 1995, pp. 455-467.

[Leveson95] Nancy G. Leveson, *Safeware*, Addison Wesley, Reading, Mass., 1995.

[Nagle82] Phyllis Nagle and James A Skrivan, "Software Reliability: Repetitive Run Experimentation and Modeling", NASA CR-165836, February 1982.

[RTCA92] "Software Considerations in Airborne Systems and Equipment Certification", RTCA SC-167, Washington, DC, 1992.

[SoHaR96] *Availability Modeling Of Air Route Traffic Control Centers (ARTCCs)*, technical report prepared for the Federal Aviation Administeration Communications, Navigation, and Surveillance division under contract DTFA01-93-Y-0169, subcontract 96-107, October, 1996.

[Tang92] D. Tang and R.K. Iyer, "Analysis and Modeling of Correlated Failures in Multicomputer Systems," *IEEE Trans. Computers*, Vol. 41, No. 5, May 1992, pp. 567-577.

[Tang95] D. Tang and M. Hecht, "Evaluation of Software Dependability Based on Stability Test Data," *Proc. 25th Int. Symp. Fault-Tolerant Computing*, Pasadena, California, pp. 434-443, June 1995.

[Tang95b] D. Tang and H. Hecht, *Measurement-Based Dependability Analysis for Critical Digital Systems*, SBIR NRC-04-94-061 Phase I Final Report, SoHaR Inc., May 1995.

[Tang96] D. Tang, M. Hecht, and H. Hecht, "A Methodology and Tool for Measurement-Based Dependability Evaluation of Digital Systems in Critical Applications", *IEEE Transactions on Nuclear Science*, August, 1996.

[Trivedi82], K. Trivedi, *Probability and Statistics with Reliabiltiy, Queuing, and Computer Science Appplications*, Prentice Hall, Englewood Cliffs, NJ, 1982

[TVA90] TVA Letter to NRC Dated May 10, 1990, *Sequoiyah Nuclear Plant (SQN) — Eagle 21 Functional Upgrade Commitments*, NRC Public Document Room, Accession #910715001.

[UKHSE87] United Kingdom Health and Safety Executive, *Programmable electronic systems in safety related applications: General technical guidelines*, London: HM Stationery Office, 1987.

[Wood96] Alan Wood, "Predicting Software Reliability", *IEEE Computer*, November, 1996, pp. 69-77.