Proceedings of the 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, University Park, PA, May 6-9, 1996, pp. 535-542.

Measurement-Based Dependability Evaluation for Safety-Grade Digital Systems[1]

Dong Tang, Myron Hecht, Herbert Hecht
SoHaR Incorporated, 8421 Wilshire Blvd., Suite 201, Beverly Hills, CA 90211

Robert Brill
Nuclear Regulatory Commission, T10-E33, Washington DC 20555

ABSTRACT

This paper presents a measurement-based methodology for dependability[2] evaluation of safety-grade digital instrumentation and control (I&C) systems. This methodology uses failure data from operational systems to assess reliability and availability, to identify problem areas, and to determine the no-failure test duration required for achieving a certain level of dependability. The methodology is used with actual data from nuclear safety, nuclear control, and air traffic control systems to demonstrate an experimental approach to the objective evaluation of dependability for critical systems.

I. INTRODUCTION

The need to replace obsolete and obsolescent I&C systems in licensed nuclear facilities has long been apparent. The industry as well as regulators find it difficult to predict the reliability of digital systems by criteria comparable to those accepted for predicting the reliability of analog systems. Static methods (including "proof of correctness") can, under the best circumstances, show freedom from a restricted class of problems, but they offer no comprehensive, quantitative prediction of run-time dependability under highly varied situations in which safety systems must operate.

There are two conventional approaches to reliability and availability prediction: (1) modeling of a system in the design phase, or (2) assessment of the system in a later phase, typically by testing. The first approach relies on probabilistic models that use component level failure rates published in handbooks or supplied by the manufacturers. This approach provides an early indication of system dependability, but the model as well as the underlying data later need to be validated by actual measurements. The second approach typically uses test data and reliability growth models. It involves fewer assumptions than the first, but it can only address the reliability growth of the target software in the development process and cannot be used to evaluate overall system operational availability. This is because software reliability growth models are based upon fault rate [5], and they do not furnish creditable predictions when there are few observed failure data, as must be the case for safety systems. In addition, these models typically do not account for recovery provisions and coverage (to be explained later) considerations.

Thus, there is a need for an alternative, or possibly a complement, to these approaches, and this is offered by the measurement-based dependability evaluation [3, 8]. The measurement-based approach is appropriate for the operational environment, but it can also be applied during testing. The primary advantage of this approach lies in use of measurements and models for interpretation of the measurements. Measurements can be performed on commercial grade components without requiring the vendor to reveal proprietary information. Based on the measurements, the approach can provide assessments of various dependability measures (e.g., failure rate, reliability, availability) with stated confidence levels.

This methodology was developed on the basis of advanced techniques in computer hardware/software failure measurement and probabilistic analysis. The methodology consists of feasible methods in data collection and processing, statistical analysis, and dependability modeling. It not only allows assessments of system dependability based on a small number of failures, but it also allows an evaluation of system dependability lower bounds at a specified confidence level where no failures were observed in the measurement period. In this paper, applications of the methodology are illustrated with operational or test data collected from three real systems used in nuclear power and air traffic control. How this methodology can be applied to address issues in evaluating safety systems is discussed in [9].

Fig. 1 shows the procedure for measurement-based analysis of operational systems. The procedure consists of three steps: (1) measurements, (2) data processing, and (3) statistical analysis and dependability modeling. The statistical analysis block can produce direct results (failure rate, etc.) for the entire system of interest, or for individual components. The latter can then be used as parameters in the subsequent dependability modeling, permitting assessment for the entire system. This modeling also allows use of the measurements to derive assessments for a different configuration.

The topics involved in the three steps (Fig. 1) will be discussed in the rest of the paper as follows: Steps 1 and 2 are covered by Sec. II, data collection and processing; Step 3 is covered by Sec. III, statistical analysis methods, and Sec. IV, dependability modeling and evaluation. The discussion will demonstrate the feasibility of the methodology with its applications on real digital systems.

[1]This research was supported by the U.S. Nuclear Regulatory Commission (NRC) under Contract NRC-04-94-061. The opinions and viewpoints expressed herein are the author's personal ones and they do not necessarily reflect the criteria, requirements, and guidelines of the NRC.

[2] The concept "dependability" was proposed at the *15th International Symposium on Fault-Tolerant Computing* in 1985 [4]. Dependability is defined as the "quality of the delivered service such that reliance can be justifiably placed on this service." The dependability impairments are faults, errors, and failures. The means to achieve dependability is through fault avoidance and fault tolerance. Two major measures of dependability are reliability and availability.

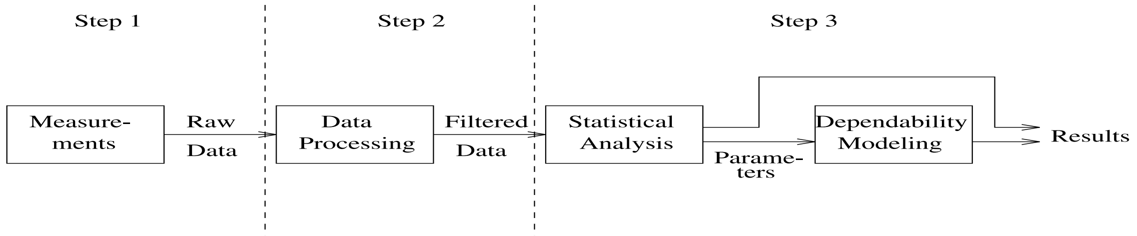

Fig. 1. Measurement-Based Analysis of Operational System

II. DATA COLLECTION AND PROCESSING

Data collection and processing lay the foundation of a measurement-based analysis of system dependability and have great impact on the subsequent analysis and results. This section discusses approaches to data collection and processing.

A. Data Collection

Dependability data can be collected from a system during testing or normal operation. In either case, the measured system should be exposed to a wide range of usage conditions for the results to be representative. Statistically significant evaluations require a considerable amount of data. In modern digital systems, especially if they are fault tolerant, failures of the entire system are infrequent. In order to obtain meaningful data, measurements need to be made for a long period of time whenever possible. Where there are multiple systems, they can be measured concurrently to accumulate the required operating time in a shorter calendar period. In addition, when testing a system, acceleration techniques can be used to reduce the testing time (e.g., test cases that stress the software by including many exception conditions). But determination of acceleration factors may be a problem. If information obtainable from the data is not sufficient for estimating the parameters required for dependability modeling, statistical inference techniques and assumptions based on experience have to be used.

There are two mutually complementary approaches to obtaining basic data: on-line automatic logging and manual logging. In the first method, an event-logging program records information on errors occurring in the system, as well as other system events such as reboots and shutdowns. Manual logging is the most frequently used means of reporting problems. Manually generated structured trouble reports have been widely used in system testing and field operation.[3] A structured trouble report usually contains information on the problem location, symptom, effect, and occurrence date. The cause of and the resolution to the problem may be later added to the report after the problem is diagnosed and resolved. An ideal environment should provide both event logs and structured trouble reports. In most cases, only one kind of data is available for analysis. Sometimes the available data are even unstructured trouble reports.

[3]The term Program Trouble Report (PTR) is also in wide use and will be used to represent the structured trouble reports in the data presented in this paper.

Three sets of data (Table 1) are used here to illustrate data collection and processing methods and will be used later to illustrate statistical analysis and dependability modeling. It should be pointed out that these data are used for the illustration purpose and they may not reflect the full range of possible input values. The data were collected from three critical digital I&C systems:

(1) The Eagle 21 system which is a microprocessor based protection system installed in the Sequoyah Nuclear Plant [11].

(2) An Integrated Control and Safety (ICS) system which is based on the digital technology to replace the analog technology being used in a group of nuclear plants.

(3) An advanced Air Traffic Control (ATC) system under development by FAA for collecting, processing, and displaying air traffic data [7].

For the Eagle 21 system, the data contained 71 incidents in unstructured letter form, typically describing the event and the action taken during operation. For the ICS system, the data were structured trouble reports covering 113 hardware and software events occurring in an early phase of system testing. For the ATC system, the data were collected from two stability tests (referred to as *Checkpoint 3* and *Checkpoint 4*) on large configurations, during a middle phase of system testing. Both event logs and PTRs were available for the two data sets with 92 and 25 software failures, respectively. More detailed descriptions of the measured systems and information about the data are provided in [8].

B. Data Processing

Data processing consists of two major activities: (1) identify failures from raw data and classify failures by some criteria, and (2) generate formatted data to facilitate subsequent analyses. Step (1) depends largely on the software tools used to capture failure events and their precursors, and only a few remarks are offered here. Step (2) is discussed in more detail later.

Many failures are obvious, such as: when processing stops; when warning messages are generated for attempted access to non-existent or protected memory; exceeding watch-dog timer limits; data communication problems. These are captured by most commercial data loggers, but it is difficult to establish that all failures (including those furnishing incorrect results) will trigger the failure detection mechanisms. Most responsible developers of software for safety systems will place assertions (of expected data values or sequence of operation) into the code, and violation of these assertions transforms a subtle, initially undetectable failure, into a

Table 1. Summary of Three Data Sets

System	Application	Data Format	No. of Events	System Phase	Collection Period
Eagle 21	Nuclear Power Safety	Unstructured Trouble Report	71	Field Operation	3 System Years
ICS System	Nuclear Power Control & Safety	Structured Trouble Report	113	System Test	1 Calendar Year Simulation Test
ATC System	Air Traffic Control	Event Log PTR	92, 25	System Test	33 & 78 Hours Stability Test

detectable one. Verification of the correctness and completeness of the assertions is one of the objectives of software test. But any methodology that makes use of observed failure data has limitations with regard to failures that are not detected by either hardware/software or the human observer.

Trouble reports and operator maintenance logs identify failures explicitly. However, screening may be required to eliminate duplications and reports that are later found not to represent failures. On-line event logs usually contain a large amount of redundant, irrelevant, and minor problem information, and they require data reduction to extract useful failure information.

There are many ways to classify failures. A failure can be permanent or transient in nature, and can be caused by hardware or software problems. The extent or severity of a failure can be divided into the component, channel,[4] and system levels. In addition, failures can be detected during normal operation or during maintenance, and thus a classification by detection mode may be desirable. The following criteria are commonly used in failure classification:

- by location in which the failure occurs
- by function affected by the failure
- by extent or severity of the failure
- by cause of the failure

Different classifications serve different purposes. The location and severity information is useful for estimating parameters for dependability modeling and assessment. The function and cause information can be used in debugging and project management. The criteria used in failure classifications depend largely on data which may not consistently provide information required by a particular criterion. If different failure reports in the same data set provide different types of information, two or more criteria have to be used in the classification.

Tables 2 through 4 list failure classifications for the three sets of data. In Table 2, the category "skinware" represents failures directly introduced by personnel. Incorrect operator actions can cause hardware or software failures, as shown in Table 3 where a separate category "skinware" is not used because these failure types have been assigned to either hardware or software problems in the corresponding trouble reports. The classification of the ICS data uses two criteria due to the limitation of the data (Table 3). This set of data represents an early phase of test, and

[4]Here a channel means a set of components required to perform the specified functionality. In a critical system, there are typically several redundant channels to provide fault tolerance.

Table 2. Failure Classification for Eagle 21 Data

Category	Classification	Criteria
Hardware	Processor, I/O, Power, Bus Memory, Unknown	Location
Software	Algorithm, Initialization, I/O Driver	Function
Skinware	Installation, Operation	Cause

Table 3. Failure Classification for ICS System Data

Category	Classification	Criteria
Hardware	Algorithm, Board, Cable, Configuration Display, Functionality, Operation Parameter, Performance, Power, Wiring	Cause Function
Software	Algorithm, Diagnosis, Display Functionality, Interface, Operation Parameter, Performance, Simulator, Timing	Cause Function

Table 4. Failure Classification for ATC System Data

Category	Classification	Criteria
Software	SW Units: BS, CC, CIP, EFC, HFC, FDM	Location
Software	AS (single copy), OU (all copies)	Severity

thus problems were identified for debugging, not for assessment. For the ATC system (Table 4), only software failure data is available. Two classifications are made for these software failures: by location and by severity. In the classification by severity, the Address Space (AS) failure is an individual failure of a software process with a successful takeover by a standby process. The Operational Unit (OU) failure is a failure of all processes in the group, which contributes to unavailability. This classification will be used for dependability modeling as discussed later.

III. STATISTICAL ANALYSIS METHODS

In this section, several statistical methods commonly used for analyzing data collected from operational systems are introduced. Applications of these methods are illustrated with the three sets of data discussed in Sec. II.

A. Basic Statistics

Basic statistics of the measured data can be presented as event absolute frequency, relative frequency (percentage), and occurrence probability, given certain environmental conditions. These statistics provide an overall picture of the system under observation. Often, dependability bottlenecks and failure trends can be identified by analyzing these statistics. Fig. 2 and 3 show basic statistics for the ICS hardware and software PTRs. It is seen that functionality is the number one software problem reported during the testing (Fig. 3), while wiring and parameter setting are the most frequent hardware problems (Fig. 2).

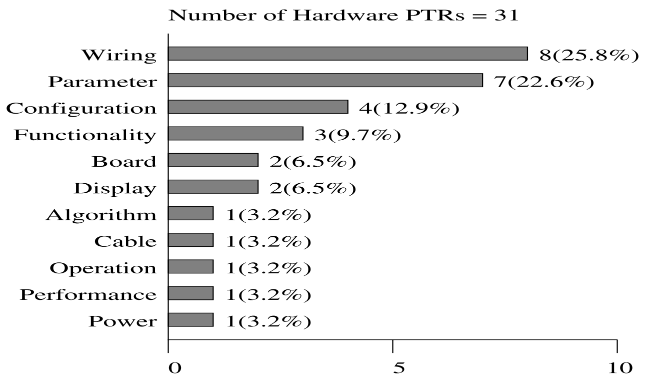

Fig. 2. Basic Statistics for ICS Hardware PTRs

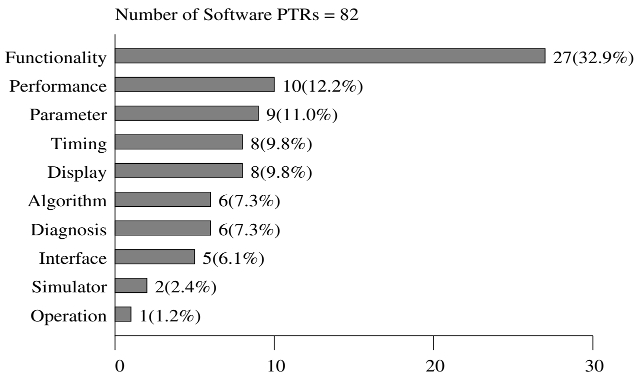

Fig. 3. Basic Statistics for ICS Software PTRs

If the measurement period is partitioned, failure statistics can be calculated for each subperiod. Changes in these statistics can provide insight into the failure reduction achieved by debugging. Table 5 lists the failure types which changed significantly from the first 6 months to the second 6 months for the ICS software trouble reports. These data relate to an early phase of test and are not typical of those normally considered for dependability assessments. Nevertheless, use of the measurement-based approach permits interesting insights to be obtained.

Algorithm and parameter problems were frequently reported in the first 6 months, but disappeared in the second 6 months. Instead, timing and diagnosis problems appeared in this later subperiod. This kind of change is typical in software testing and can be used as an indicator of software maturity. In the early period of testing, the obvious problems (algorithm and parameter problems) are exposed. In the later period of testing, problems that occur in rare conditions (timing and difficult diagnostic conditions) are exposed. As the testing continues, other kinds of problems may be exposed. Since the data discussed here only represent an early segment of testing, the problems identified are more an indicator of debugging activities, rather than a quantification of system dependability.

Table 5. ICS Software PTR Type Trends

PTR Type	First 6 Months	Second 6 Months
Algorithm	6	0
Parameter	9	0
Timing	0	8
Diagnosis	0	6

B. Estimation of Mean Time Between Failures (MTBF)

Assume that the random variable TBF (Time Between Failures) is exponentially distributed with a mean of M. If n failures have been observed during the measurement period T, then the sample mean

$$\bar{X} = \frac{T}{n} \tag{1}$$

is an estimate of M. Further, a confidence interval of M can be estimated using the following equation [3]:

$$\frac{2T}{\chi^2_{1-\alpha;2n}} \le M \le \frac{2T}{\chi^2_{\alpha;2n}} \tag{2}$$

where α is the significance level ($1-\alpha$ is the confidence level), and $\chi^2_{\alpha;2n}$ and $\chi^2_{1-\alpha;2n}$ are two values from the Chi-square distribution table such that the random variable X^2_{2n} satisfies $Prob(X^2 < \chi^2_{\alpha;2n}) = \alpha$ and $Prob(X^2 < \chi^2_{1-\alpha;2n}) = 1-\alpha$.

Reliability Growth Analysis

The Eagle 21 data is appropriate for demonstrating reliability growth analysis. The whole data collection period (2 calendar years) for Eagle 21 is divided into four subperiods, with each being approximately a half year. For each subperiod, the mean and the 90% confidence interval of MTBF are calculated using Eq. (2) and Eq. (3). The results are listed in Table 6.

Table 6. MTBF by Subperiod (half-year) for Eagle 21

Subperiod	1	2	3	4
Lower Limit	8.2	7.2	32.8	19.9
Mean	11.3	9.3	55.4	38.5
Upper Limit	16.7	12.4	118	113

There is no statistical difference between subperiods 1 and 2, and between subperiods 3 and 4, because the mean of both periods (1 and 2, or 3 and 4) falls into the other's confidence interval. Although the means for subperiods 3 and 4 differ considerably (55.4 vs. 38.5), the confidence intervals for the two subperiods show that their MTBFs are not statistically different. The large confidence interval for subperiod 4 is due to the small sample size of the subperiod. However, reliability growth is obvious from the first year to the second year, because the confidence intervals for subperiods 1 and 2 are out of range of those for subperiods 3 and 4.

C. Estimation of Failure Rate Upper Bound

Failure rate is frequently a required parameter in dependability modeling. For critical systems, it is desired to use upper bounds of failure rates (instead of means) for the modeled components, with a certain level of confidence. Given a sample of observations on a hardware or software unit for a time period of T, there are two possible situations:

(1) At least one failure has been observed during T;

(2) No failure has occurred in T.

In either situation, an upper bound for the failure rate, λ_u, can be estimated from data with a certain level of confidence.

In situation (1), λ_u can be derived from Eq. (2):

$$\lambda_u = \frac{\chi^2_{1-\alpha;2n}}{2T} \tag{3}$$

where $1-\alpha$ is the specified confidence level, and $\chi^2_{1-\alpha;2n}$ can be found from the Chi-square distribution table.

In situation (2), λ_u can be estimated by [8]

$$\lambda_u = \frac{-ln(\alpha)}{T} \tag{4}$$

These two equations can be used to estimate failure rate upper bounds from either operational or test data. Table 7 shows results of the estimation process for selected OUs/FGs (basic software units of the ATC data set) using the ATC Checkpoint 4 data at the 80% confidence level. In the table, the column of "OU/FG" is the name of the tested software unit, the column of "No. of Copies" is the number of software copies which were actively running in the test, and "Operating Time" is the product of the number of copies and the test duration.

Table 7. Failure Rate Estimation for ATC System Data

OU/FG	No. of Copies	Operating Time (hr)	No. of Failures	Failure Rate Upper Bound
BS	87	6812.1	0	2.36×10^{-4}
CCP/CCS	87	6812.1	2	4.40×10^{-4}
CIP	56	4384.8	4	1.26×10^{-3}
EFC	12	939.6	10	1.33×10^{-2}
FDM	12	939.6	1	1.71×10^{-3}
HFC	12	939.6	0	1.71×10^{-3}

D. Estimation of Coverage

In a fault tolerant system, most component or subsystem failures can be covered (prevented from causing external effects) with hardware, software, or combined fault tolerance approaches. However, some failures are not covered and manifest themselves to higher levels, or even to the external level, due to defects in the fault tolerance implementation. For a given system level, the ratio between the number of covered failures and the number of total failures is referred to as *coverage*, denoted by C [1]. Previous research has shown that coverage is a very sensitive parameter in dependability modeling of fault tolerant systems [2].

Estimation of coverage is often complicated by the lack of non-covered failures (the number of non-covered failures is zero). In such a case, the estimated mean coverage would be 1 (a 100% coverage), which may not be representative of the likely true state because of various reasons such as imperfect error detection mechanisms, defects in redundancy management software, and common mode failures. Thus, a value less than 1 is used in conservative dependability modeling. Similar to the estimation of failure rate upper bound, a lower bound of coverage, C_l, can be derived for the situation in which non-covered failures were not observed during the measurement period:

$$C_l = \alpha^{\frac{1}{n}} \tag{5}$$

where n is the total number of failures which were covered (as observed in the test or operation), and α is the desired significance level ($1-\alpha$ is the confidence level). Derivation of this equation is given in [8].

When n is not large enough (below 30), the estimated C_l is less than 0.95 which is a default value used in dependability modeling at the design phase. Thus, if C_l is greater than 0.95, C_l is used. Otherwise, 0.95 is used. Table 8 shows the coverage estimation process and results for selected OUs/FGs using the ATC Checkpoint 3 data. The 80% confidence lower bounds are all lower than 0.95. As the accumulated number of covered failures increases, it is possible for the lower bound to exceed 0.95.

Table 8. Coverage Estimation for ATC System Data

OU/FG	Total No. of Failures	No. of Non-covered Failures	80% Confidence Lower Bound	Coverage Estimate
BS	10	0	0.85	0.95
CCP/CCS	22	2		0.91
CIP	15	0	0.90	0.95
EFC	4	0	0.67	0.95
FDM	11	4		0.64
HFC	5	1		0.80

IV. DEPENDABILITY MODELING & EVALUATION

A dependability model is a mathematical model that characterizes a system with a certain level of abstraction. The structure of the model reflects the architecture or behavior of the modeled system. The parameters of the model are either dependability measures (i.e., failure rate, recovery rate, etc.) of components of the modeled system, or correlation measures characterizing relationships between the components (i.e., coverage, conditional failure probability, etc.). The model outputs are system dependability measures (i.e., availability, reliability, etc.).

Dependability models can be used to (1) assess the current availability, reliability, and performability, (2) identify problem areas that most impede the current dependability, and (3) predict the required test or operational time for achieving a certain level of dependability. In all cases the methodology discussed here will extract more information from the available data than conventional statistical approaches. This section discusses the model structures and demonstrates their applications using the data introduced in Sec. II.

A. Model Structures and Modeling Approaches

Three model structures have been found useful in measurement-based dependability modeling: the reliability block diagram, the K-out-of-N model, and the Markov chain [10]. A reliability block diagram is a graphical method of depicting the components that can be connected in serial, parallel, or their combination. A series system requires all the components to be functioning, while a parallel system requires only one of the components to be functioning, for the system to function properly. In a K-out-of-N model, the system consists of N components, K of which are required to be functioning for the system to function properly. A Markov chain uses system states and state transitions to model various combinations of operational and failed components in the system (represented by states) and possible transitions from one combination to another. Both reliability diagrams and K-out-of-N models are combinatorial models and typically assume failure independence among modeled components. Markov chains are stochastic models which can incorporate interactions among components and failure dependence in the model.

If failures of components are relatively independent in the modeled system, the simpler structures of reliability block diagrams and K-out-of-N models suffice. This is usually the case for the high level modeling in which lower level failures can be considered independent. If in the modeled system, failure of a component may affect other components, or a reconfiguration is involved upon a component failure, Markov chains are preferred. This is usually the case for the low level modeling or any modeling where failure interactions among components are strong.

A system can be modeled hierarchically: system level, subsystem level, and lower levels of modeling as required. This hierarchical modeling approach is recommended because 1) it reduces model complexity and 2) it facilitates identifying problem areas (by comparing results from submodels). Different levels of modeling may use different model structures, depending on the architecture and behavior of the modeled system. In the next subsection, the hierarchical modeling approach and the three model structures are illustrated by developing models for the ATC Sector Suite Subsystem. In the following subsection, the models are used to demonstrate three applications of availability modeling.

B. Illustration of Dependability Modeling

This subsection develops software dependability models for the ATC Sector Suite Subsystem. The models are developed at three levels: system model, group model, and subgroup model. The system model is a K-out-of-N model. Each group model is a reliability block diagram. Each subgroup model is a Markov chain.

At the system level, the modeled system consists of 50 groups, or sector suites, and is assumed to allow a loss of at most one group. Thus, a 49-out-of-50 model can be applied to the system level. The 49-out-of-50 model requires either all 50 groups or any 49 of the 50 groups to be available for the system to be functioning. Let A_g denote availability for a group. Then the system availability, A, can be calculated by

$$A = {A_g}^{50} + 50 \times {A_g}^{49} \times (1 - A_g) \tag{6}$$

At the group level, there are two types of software units in a sector suite: the application software (OUs) and the system support software (FGs). Thus, the group model has two components: the application subgroup and the system support subgroup. Fig. 4 shows the two blocks connected in series. If failures in the two blocks are assumed independent, the group availability is the product of the availabilities of the two subgroups:

$$A_g = A_{app} \times A_{sys} \tag{7}$$

where A_{app} is the availability of the application subgroup and A_{sys} is the availability of the system support subgroup.

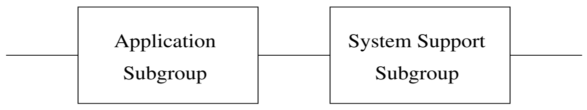

Fig. 4. The Sector Suite Group Model

Because reconfigurations are involved in the failure recovery process for both subgroups, each of them should be modeled separately by a Markov chain. Fig. 5 shows the application subgroup model. The modeled software units are three application OUs: EFC, FDM, and HFC. All of these OUs are considered necessary for the subgroup to function properly. Each OU is assumed to consist of a PAS (Primary Address Space) and a single SAS (Standby Address Space). Normally, only the PAS is actively running in the test. When the PAS fails, the SAS will take over if the reconfiguration is successful.

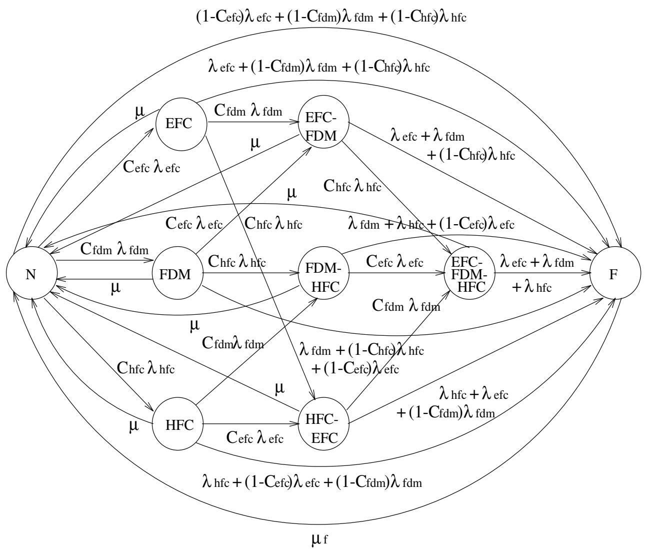
(1-Cefc)λ efc + (1-Cfdm)λ fdm + (1-Chfc)λ hfc
λ efc + (1-Cfdm)λ fdm + (1-Chfc)λ hfc
μ
EFC
Cfdm λ fdm
EFC-FDM
μ
λ efc + λ fdm + (1-Chfc)λ hfc
Cefc λ efc
Chfc λ hfc
μ
Cefc λ efc
Chfc λ hfc
λ fdm + λ hfc + (1-Cefc)λ efc
Cfdm λ fdm
N
FDM
Chfc λ hfc
FDM-HFC
Cefc λ efc
EFC-FDM-HFC
λ efc + λ fdm + λ hfc
F
μ
Cfdm λ fdm
μ
Cfdm λ fdm
λ fdm + (1-Chfc)λ hfc + (1-Cefc)λ efc
Chfc λ hfc
μ
λ hfc + λ efc + (1-Cfdm)λ fdm
μ
HFC
Cefc λ efc
HFC-EFC
λ hfc + (1-Cefc)λ efc + (1-Cfdm)λ fdm
μ f

Fig. 5. The Application Subgroup Model

The model consists of 9 states designated as circles. These states are connected by a number of allowable transitions. The notation for the states and transitions is explained as follows:

N — normal state in which all OUs are functioning properly

F — failure state in which the group is unable to provide required service

x — state in which x is recovering from a PAS failure

x_1-x_2 — state in which x_1 and x_2 are recovering from PAS failures

x_1-x_2-x_3 — state in which x_1, x_2, and x_3 are recovering from PAS failures

λ_x — failure rate of x's PAS

C_x — coverage for x

μ — recovery rate for a PAS failure

μ_f — recovery rate for an OU failure

where x (and x_1 to x_3) can be EFC, FDM, or HFC.

A complete description of Markov modeling is out of the scope of this report. However, this paragraph provides some guidance on how to interpret the diagram. The state at the extreme left is a sector suite with all three OUs running and is designated as "N" (normal). The EFC's PAS may fail which could result in two possibilities: (1) a successful OU recovery in which the PAS is replaced by the SAS, and (2) an unsuccessful recovery in which the transition from the PAS to the SAS fails and the sector suite itself is down as a result. The successful recovery state is designated by the label "EFC". The rate at which this transition occurs is λ_{fdm} multiplied by the probability of a successful transition, C_{efc} (coverage). The transition to the unsuccessful recovery state, F, is depicted as a large arc at the top of the diagram and occurs at the rate $(1 - C_{efc})\lambda_{efc}$. Because this same transition can also occur if FDM or HFC fails, the arc shows that the total transition rate is the sum of all terms representing these failures.

As explained above, the failure dependence between AS's inside an OU is modeled by the coverage parameter. However, the model assumes failure independence between OUs. This is basically supported by the data. No simultaneous OU failures were found from the data. In the Checkpoint 3 data, there were cases in which two different AS failures occurred closely in time (in the order of seconds). But they have no impact on results because these individual failures do not contribute to unavailability.

The system support subgroup is represented by a similar Markov chain. Details are discussed in [8].

C. Illustration of Dependability Evaluation

The models developed in the previous subsection and the parameters estimated in Section III allow us to evaluate software availability and other dependability measures for the sector suite subsystem. In this subsection, evaluation is demonstrated for three purposes: (1) estimation of current availability, (2) quantification of possible availability improvement if key problems reported in a few trouble reports are resolved, and (3) prediction of the possible range of availability lower bounds that can be reached through stability tests with reasonable test durations. The solutions were obtained by use of SHARPE [6], a dependability modeling tool.

Estimation of Current Availability

Table 9 lists software unavailability evaluated based on the failure rate upper bounds estimated at the 80% confidence level, for Checkpoints 3 and 4. In Checkpoint 3, the system support software dominates the group availability. In Checkpoint 4, both subgroups contribute to the group unavailability equally, after the availability of the system support software has been greatly improved. The sector suite group availability and system availability are both significantly improved from Checkpoint 3 to Checkpoint 4. An *improvement factor* from test i to test j can be defined to quantify the improvement:

$$f(i, j) = \frac{U_i}{U_j} \tag{8}$$

where U_i is the unavailability evaluated from test i, and U_j is the unavailability evaluated from test j. By this measure, the improvement factor is 17.7 for the sector suite group availability, and 30.7 for the sector suite system availability, from Checkpoint 3 to Checkpoint 4, as shown in the table.

Table 9. Unavailabilities Evaluated for Checkpoints 3 & 4

Test	Application Subgroup	Sys. Support Subgroup	Sector Suite Group	Sector Suite System
Ckpt 3	4.50×10^{-5}	6.36×10^{-4}	6.81×10^{-4}	5.56×10^{-4}
Ckpt 4	1.90×10^{-5}	1.95×10^{-5}	3.84×10^{-5}	1.81×10^{-6}
$f(3,4)$	2.4	32.6	17.7	30.7

Quantification of Availability Bottlenecks

Based upon the type of data obtained from the ATC system, the approach discussed can be used to estimate the current availability as shown above. This approach can also identify the bottlenecks in the current availability and estimate the availability improvement if the bottlenecks are removed. One way to identify such bottlenecks is to analyze model sensitivity to different trouble reports. The Checkpoint 4 data are used to illustrate the analysis.

By observing the coverage parameters for Checkpoint 4 it can be found that there are two low coverage parameters: 0.5 for EFC and 0.75 for CIP. The failure instances responsible for these low coverages are attributed to problems reported in two particular PTRs. Significant improvement could be made by resolving the problems. To quantify the possible improvement, all failure instances related to the two PTRs were taken from the Checkpoint 4 data. The reduced data set represents a hypothetical test and it is referred to as *Checkpoint 4'*. The results evaluated from the Checkpoint 4' data are shown in Table 10. For comparison, the Checkpoint 4 results are also listed in the table. The application software could be improved by a factor of 26.6 from Checkpoint 4 to Checkpoint 4'. For the overall sector suite software system availability, the improvement factor is 4.9.

Determination of Availability under Various Test Durations

The results evaluated for Checkpoints 3 and 4 are based on the stability tests of 33 and 78 hours, respectively. The number of

Table 10. Unavailability Predicted for Checkpoint 4'

Test	Application Subgroup	Sys. Support Subgroup	Sector Suite Group	Sector Suite System
Ckpt 4	1.90×10^{-5}	1.95×10^{-5}	3.84×10^{-5}	1.81×10^{-6}
Ckpt 4'	7.13×10^{-7}	1.67×10^{-5}	1.74×10^{-5}	3.71×10^{-7}
f(4,4')	26.6	1.2	2.2	4.9

failures observed was significantly reduced from Checkpoint 3 to Checkpoint 4 (from 92 to 25). As the software quality is continuously improved, the observable failures in a limited test duration can disappear. A question from the viewpoint of the project management is: How many no-failure test hours are necessary for reaching an objective availability at a desired confidence level? The curves shown in Fig. 6 address this issue. To obtain the points on the curves, the failure rates were estimated assuming no failure occurred during the test, at the 80% confidence level. The system configuration and the failure recovery rates used were the same as those for checkpoint 4.

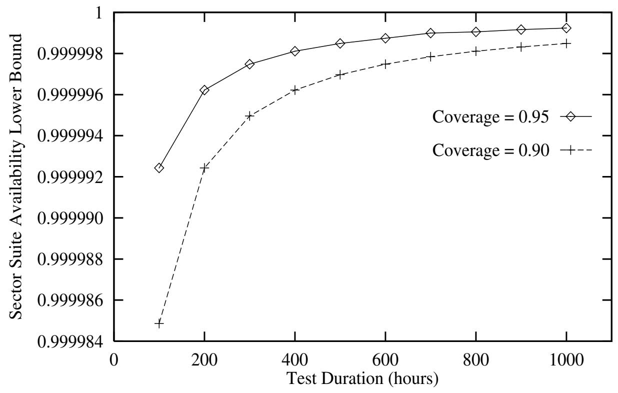

Fig. 6. Availability Achieved for Various Test Durations

As is evident in the figure, a no-failure test duration of 100 hours can demonstrate an availability of five 9's (0.999992) with an 80% confidence if the coverage is assumed to be 0.95. If the coverage is reduced to 0.90, the required test hours for the same availability at the same confidence level are increased to 200 hours. The availability grows fast when the test duration increases from 100 to 200 hours. As the test duration further increases, the availability growth slows down. For the coverage of 0.95, it takes approximately 800 hours for availability to reach six 9's (0.99999905). For the coverage of 0.90, 1000 hours of test are not sufficient for reaching this level of availability.

V. CONCLUSION

This paper has discussed a measurement-based dependability evaluation methodology for critical digital systems. The methodology consists of methods in data collection and processing, statistical analysis, and dependability modeling and evaluation. Applications of the methodology were demonstrated with operational or test data collected from three real systems used in nuclear power and air traffic control. Examples of these applications are: reliability growth analysis, assessment of system availability, identification of problem areas, and determination of no-failure test duration required for achieving a certain level of availability. How the methodology is used to model safety systems and is implemented in a software tool can be found in [9].

REFERENCES

[1] T.F. Arnold, "The Concept of Coverage and Its Effect on the Reliability Model of a Repairable System," *IEEE Transactions on Computers*, Vol. 22, No. 3, pp. 251-254, March 1973.

[2] J.B. Dugan and K.S. Trivedi, "Coverage Modeling for Dependability Analysis of Fault-Tolerant Systems," *IEEE Transactions on Computers*, Vol. 38, No. 6, pp. 775-787, June 1989.

[3] R.K. Iyer and D. Tang, *Experimental Analysis of Computer System Dependability*, to be included in *Fault-Tolerant Computer System Design*, D.K. Pradhan (ed.), Prentice-Hall, 1995.

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

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

[6] R.A. Sahner and K.S. Trivedi, "A Software Tool for Learning About Stochastic Models," *IEEE Trans. Education*, Vol. 36, No. 1, pp. 56-61, Feb. 1993.

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

[8] 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.

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

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

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