September 1990

UILU-ENG-90-2243
CRHC-90-10

Center for Reliable and High-Performance Computing

REPORT OF THE IEEE WORKSHOP ON MEASUREMENT AND MODELING OF COMPUTER DEPENDABILITY

Ravishankar K. Iyer
Daniel P. Siewiorek
Editors

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
UILU-ENG-90-2243 CRHC-90-10

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Coordinated Science Lab
University of Illinois

6b. OFFICE SYMBOL *(If applicable)*
N/A

7a. NAME OF MONITORING ORGANIZATION
NASA Langley Research Center
Hampton, VA 23665

6c. ADDRESS *(City, State, and ZIP Code)*
1101 W. Springfield Ave.
Urbana, IL 61801

7b. ADDRESS *(City, State, and ZIP Code)*
NASA Langley Research Center
Hampton, VA 23665

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
NASA

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
NASA NAG-1-602 NASA NAG-1-613

8c. ADDRESS *(City, State, and ZIP Code)*
NASA Langley Research Center
Hampton, VA 23665

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Report of the IEEE Workshop on Measurement and Modeling of Computer Dependability

12. PERSONAL AUTHOR(S)
Edited by Ravishankar K. Iyer and Daniel P. Siewiorek

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	May 1 and 2, 1990	31

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	dependability, hardware, software, modeling, measurement, data collection, statistical analysis

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A workshop was held at The Aerospace Corporation facility in El Segundo, California on May 1 and 2, 1990. The purpose of the workshop was to promote interaction among modelers and experimentalists. Participation was solicited to cover all aspects of experimentation and analysis of computer dependability including, but not limited to 1) design of experiments; 2) modeling and analysis; 3) instrumentation and data collection; 4) measurements of reliability and performance; 5) statistical analysis and interpretation; 6) software reliability and fault tolerance; 7) evaluation of experimental systems; and 8) experimental methodology. Participants who made a presentation at this conference submitted a position statement in advance of the workshop. This report contains all the position statements submitted and a summary of key conclusions and recommendations resulting from this workshop.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

22b. TELEPHONE *(Include Area Code)*

22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Report of the IEEE Workshop on Measurement and Modeling of Computer Dependability

The Aerospace Corporation
El Segundo, California

Edited by:

Ravishankar K. Iyer
Center for Reliable and High Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Daniel P. Siewiorek
School of Computer Science
Carnegie Mellon University

May 1 and 2, 1990

IEEE Workshop on Measurement and Modeling of Computer Dependability

The Aerospace Corporation
El Segundo, California, USA

Workshop Chairs:

Ravishankar K. Iyer
Center for Reliable and High Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue Urbana, IL 61801
Tel: (217) 333-9732
e-mail: iyer@crhc.uiuc.edu

Professor Daniel P. Siewiorek
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
(412) 268-2570
e-mail: dps@a.gp.cs.cmu.edu

Technical Program Committee:

J. Arlat
LAAS du CNRS
France

G.M. Masson
Johns Hopkins University

K.S. Trivedi
Duke University

Contents

CALL FOR PARTICIPATION

Workshop
on
Measurement and Modeling of Computer Dependability
May 1 and 2, 1990 – Los Angeles, California

This announcement is to solicit participation in a workshop on the evaluation of computer dependability, from experimentation to analysis. Mathematicians have been modeling computer systems for over four decades. Over the last two decades, data has been collected from operational computer systems. The purpose of this workshop is to promote interaction between modelers and experimentalists. Questions we might ask include: How should experiments be designed to collect data usable by modelers? How should models be changed to reflect the realities of experimentation? Participation is solicited to cover all aspects of experimentation and analysis of computer dependability including, but not limited to the following topics.

1. Design of Experiments
2. Modeling and Analysis
3. Instrumentation and Data Collection
4. Measurements of Reliability and Performance
5. Statistical Analysis and Interpretation
6. Software Reliability and Fault Tolerance
7. Evaluation of Experimental Systems
8. Experimental Methodology

Prospective participants should send a brief outline of their interest and position statement (no longer than 200 words) to Professor Ravi K. Iyer at the address below, no later than March 1, 1990. It is expected that the attendance to the workshop will be limited to about fifty participants. All position statements will be reviewed by the technical program committee and the accepted participants will be notified by March 15, 1990.

Workshop Chairs

Professor Ravi K. Iyer
Center for Reliable and High Performance Computing
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue Urbana, IL 61801
Tel: (217) 333-9732
e-mail: iyer@crhc.uiuc.edu

Professor Daniel P. Siewiorek
School of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
(412) 268-2570
e-mail: dps@a.gp.cs.cmu.edu

Technical Program Committee

J. Arlat	LAAS	France
G.M. Masson	Johns Hopkins	USA
K.S. Trivedi	Duke University	USA

1 List of Position Statements for Workshop

1. Gary Engel, UNISYS, "Analysis and prediction of system reliability."
2. Ed Upchurch and John Peterson, JPL, "Discrete event hypercube models used to evaluate performance trade-offs and understand actual performance anomalies."
3. Dale Lomelino, Honeywell, "Fault tolerance evaluation and design."
4. Celeste Belcastro, NASA-Langley, "Evaluation of fault-tolerant control systems in harsh electromagnetic environments."
5. Johan Karlsson, Chalmers University, Sweden, "Fault injection system for the study of transient fault effects."
6. Bjarne E. Helvik, The University of Trondheim, Norway, "Operational failures in SPC systems."
7. Nancy Leveson, "Experimental methodology."
8. Israel Koren, University of Massachusetts, "Modular and dependable multiprocessing systems."
9. Ann Patterson-Hine, Joanne Bechta Dugan and Jerry Yan, NASA–Ames, "Intelligent computational systems for space missions."
10. Gerald M. Masson, Johns Hopkins University, "Hardware-Based fault injection systems."
11. Ambuj Goyal, IBM, "Simulating Markovian models of dependable systems."
12. Ram Chillarege, IBM, "Relating software defect types to the reliability growth experienced."
13. Laureen Brickley, IBM, "Spaceborne computing."
14. J.C. Laprie, C. Beounes, M. Kaaniche and K. Kanoun, LAAS du CNRS, France, "The transformation approach to the modeling and evaluation of the reliability and availability growth of systems in operation."
15. J. Arlat, K. Kanoun and J.C. Laprie, LAAS du CNRS, France, "Dependability modeling and evaluation of software-fault tolerance approaches."
16. K.H. Kim, B.J. Min and W.J. Guan, University of California–Irvine, "Experimental validation of two fundamental design techniques for fault-tolerant distributed computing."
17. Ed McCluskey, Stanford University, "Errors in operational digital systems."
18. Robert Goan, SCI, "Real time parallel processing systems for high-reliability military applications."
19. Doug Miller, George Mason University, "Statistical modeling."
20. Kang Shin, University of Michigan, "Fault-injection, testing, and measurement experiments on HARTS."
21. Kishor Trivedi, Duke University, "Interfaces between modeling and measurements."
22. Robert Geist, Clemson University, "Modeling recovery time distributions in ultrareliable fault-tolerant systems," and "Conservative reliability predictions for real-time software."
23. J.H. Lala, C.S. Draper Laboratories, "Validation of real time fault tolerant computers via measurements and modeling."
24. Dr. Michael R. Lyu, JPL, "Software reliability experiences at the Jet Propulsion Laboratory."

25. William J. Watson, Tandem, "Design of *Integrity S2*: A fault-tolerant RISC unix minicomputer."

26. Oleg Panfilov, NCR, "Predictive methodology."

27. M.C. Hsueh, DEC, "Failure data analysis."

28. Andrew Reibman, AT&T, "Practical issues in modeling system availability."

29. Dr. John Kelly, University of California–Santa Barbara, "Techniques for building dependable distributed systems: Experimental evaluation."

30. W. Kent Fuchs, University of Illinois at Urbana-Champaign, "Trace driven evaluation of fault-tolerant parallel architectures."

2 Schedule

Day 1

8:45	Panel Session on Measurements and Experiments
	Panelists: E. McCluskey, J. Lala, K. Shin and D. Miller (Panel 1).
9:30	Break into discussion groups
	Group A: E. McCluskey, J. Lala, K. Fuchs
	Group B: K. Shin, D. Miller, R. Chillarege
12:15	Lunch
1:30	Panel Session on Modeling
	Panelists: G. Masson, J. Arlat, K. Trivedi and R. Chillarege (Panel 2).
2:15	Break into discussion groups.
	Group A: D. Masson, J. Arlat and J. Dugan.
	Group B: K. Trivedi, R. Chillarege and A. Reibman.
4:45	Concluding Remarks for day

Day 2

8:30	Bridging the Gap between Modelling and Measurement
9:00	Summary/Highlights from Group Meetings
11:30	General Discussion and Wrap Up
12:00	Adjournment

3 Position Statements

3.1 Errors in Operational Digital Systems

Ed McCluskey
Stanford University

I would like to discuss the physical sources of errors in operational digital systems and the appropriate techniques for relating these causes to the resulting errors. Of particular interest are the questions of temporary failures and the use of the single-stuck fault model as a model for them. Some other questions that I think are interesting and am prepared to discuss include the following: Are statistical techniques useful for the reliability level of interest? Is design verification adequate with existing methodologies? Will system failure events be caused by physical flaws more often than by design defects?

3.2 Validation of Real Time Fault Tolerant Computers via Measurements and Modeling

Jaynarayan H. Lala
The Charles Stark Draper Laboratory, Inc.

The overall goal of the Fault Tolerant Systems Division at Draper is to design and build validated computer systems for real time applications where human safety, the successful outcome of a mission and/or a high value asset is critically dependent on the correct and timely functioning of a computer system. In this context, it is necessary to validate the performance and reliability of the computer system. The validation approach involves a judicious combination of analytical modeling, simulations and empirical measurements of critical parameters on proof-of-concept and prototypical systems.

In particular, the requirement of extremely low system failure rates for these applications (typically $10^{-}6to10^{-}10$ per hour) precludes computer reliability validation exclusively by means of experimental evaluation. Therefore, the overall system degradation from an initial fault-free state to the final system failed state due to component failures and intermediate transitions due to repair actions are modeled analytically using Markov processes and combinatorial methods. (It should be noted, however, that the state-of-the-art still does not permit validation required for the ultra-low failure rates.) A set of Macintosh based tools has been developed that allow the designer to quickly explore the effect of various architectural options, repair strategies, *etc.* on the system reliability and availability. The models use a number of parameters whose values depend on the system behavior under fault-free and faulted conditions. Examples include time to detect, isolate and reconfigure (FDIR) around a fault. Empirical measurements on a proof-of-concept system are used to refine the estimate of these parameters.

The requirement of hard dead-lines for certain response times in these real time applications makes the use of statisical methods such as queueing theory unsuitable for performance validation. Many performance characteristics such as transport lag and CPU utilization can be calculated using deterministic methods. Other more complex characteristics such as interactions between CPU, I/O and buses need to be analyzed using Monte Carlo simulations and empirical measurements. Various operating system overheads such as scheduling and dispatching a task and redundancy management overheads such as the throughput devoted to FDIR lend themselves to empirical measurements.

Finally, empirical evaluation of computer system performance in the presence of faults ties together and validates the performance and fault tolerant characteristics of the system.

3.3 Fault Injection, Testing, and Measurement Experiments on HARTS

Kang G. Shin
The University of Michigan

Due to their potential for high reliability and throughput via the multiplicity of components, distributed computing systems are being increasingly used for reliability- and time- critical applications. However, the probability of having one or more component faliures in a distributed system increases with the number of components used. Thus, the ability of locating faulty components, isolating them, reconfiguring the system, and resuming the computation is a key to the success in realizing the potential of any distributed system. As part of our global goal of designing fault-tolerant distributed systems, we are currently developing system-level fault diagnosis algorithms for large multiprocessor/multicomputer systems.

I will describe our plan of how to implement the diagnosis algorithms and measure their performance on an experimental multicomputer system, called HARTS, which is currently being built at the Real-Time Computing Laboratory (RTCL), The University of Michigan. Specifically, I will discuss planned methods and tools (based on our past experiences with FTMP) for injecting faults, generating synthetic workloads, applying tests, measuring fault coverage, and applying the diagnosis algorithms on HARTS.

3.4 Statistical Modeling

Doug Miller
George Mason University

Much of the statistical modeling of computer dependability falls into two categories. Some models describe highly aggregated phenomena, ignoring much of the detail of the system, thereby under-utilizing data which may be available, and consequently inferring less about system behavior; an example of this is most software reliability growth models which don't distinguish among classes of errors or don't use additional information that may become available during repair. Some other models are very detailed, but are based on parameters (or assumptions such as stochastic independence) that are impossible to determine with sufficient accuracy in order to use the model; some structural models of software failure behavior seem to fall in this category. (In this case the modelers say that the models should be used qualitatively to gain conceptual understanding.) To develop good quantitative models lying between these two extremes, I (as a statistical modeler) need more interaction with computer and software engineers who understand the systems. One approach that might be fruitful is to try to extend some univariate failure models to a multivariate context; for example, multiple classes of software/hardware failures modelled as marked point processes.

3.5 Hardware-Based Fault Injection Systems

Gerald M. Masson
Johns Hopkins University

Fault injection systems provide for a unique experimental environment for accelerating the design, verification and testing of highly reliable computer systems, as well as supporting the study of various fault/error/upset models and fault monitoring/detection mechanisms. In a manner somewhat Different from other systems, our fault injection system generates physical faults mainly by hardware insertion techniques. Currently, the target system is an MC-68000 based board, but we can easily apply our injection techniques to other systems. Currently, our system provides for 224 kinds of intermittent/ transient faults including single/ multiple faults of stuck/ bridging types with fault duration varying from 250ns to 64μs. Also, the number of fault types is extendable. The fault injection commands can be arbitrarily inserted into a user's program, so it is sufficiently flexible to create a wide variety of fault conditions. The injection process is simple to control. Since the faults are generated by hardware, the effect of an injected fault in a program may differ from one injection to the next. We believe that the faults generated by our system are

closer to the intermittent/ transient faults of the real environments. Numerous experiments have been run and we have obtained experimental data relative to fault/upset coverage for faults injected in the data bus, address bus and control bus of our testbed computer systems.

3.6 Dependability Modeling and Evaluation of Software-Fault Tolerance Approaches

Jean Arlat, Karama Kanoun and Jean-Claude Laprie
LAAS du CNRS
France

This work provides a unified modeling framework for the evaluation of the dependability of three software-fault tolerance approaches, namely: recovery blocks (RB), N-version programming (NVP) and N-self-checking programming (NSCP).

The study is based on the analysis of architectures (including variants and deciders as well) able to tolerate a single fault:

- RB: two alternates and an acceptance test,
- NVP: three versions and a voter,
- NSCP: four versions and two comparators.

The main features of the presentation incorporate:

- The identification of the possible types of faults through the analysis of the software production process, which lead to account for both **independent** faults in the variants or the deciders and **related** faults between them;
- The construction of Markov models describing the operational behavior of the considered architectures: in order to reflect most of the reported observations, it has been considered that related faults manifest as **similar** errors and are thus prone to lead to **common-mode** failures, whereas we have assumed that independent faults cause **distinct** errors that may lead to **separate** failures;
- The processing of the models enabling the investigation of **safety** and **reliability** issues;
- The identification and the analysis of the major parameters that condition the **reliability improvement** achieved by each architecture with respect to a non fault-tolerant software.

This work is a continuation of the investigation presented at FTCS-18 for RB and NVP: the NSCP approach is included. This approach corresponds to the approach currently used in most real-life applications of software-fault tolerance.

3.7 Interfaces Between Modeling and Measurements

Kishor S. Trivedi
Duke University

Measurements and modeling are two complementary activities that provide insights into computer system behavior. By using a judicious combination of the two cost-effective methods of system evaluation can be devised. Two interfaces between modeling and measurements, namely model calibration and model validation, will be explored in this talk.

3.8 Relating Software Defect Types to the Reliability Growth Experienced

Ram Chillarege
IBM Thomas J. Watson Research Center

This talk presents an empirical study on software defects with a goal to gain understanding of defect types and their influence on reliability growth. The study provides insight into possible differences in characteristics among the different defect types. The finding demonstrates the usefulness of such analysis in impacting the testing strategy and the development process. The study performed on a large software project shows that:

- The population of defects is separable into sub-populations demonstrating different reliability growth characteristics.
- Initialization defects are found to be strongly related to defect sub-populations with very inflected growth curves. Thus, impacting the testing strategy.
- Function and checking type defects contribute the largest cause of defects. However, improvements in specification can reduce the incidence of these defects.

3.9 Upset Evaluation of Fault-Tolerant Control System in Harsh Electromagnetic Environments

Celeste M. Belcastro
NASA Langley Research Center

Control systems for advanced aircraft will have very high reliability specifications which must be met in adverse as well as nominal operating conditions. Severe operating conditions can result from electromagnetic disturbances caused by lightning, High Energy Radio Frequency (HERF) transmitters, and Nuclear Electromagnetic Pulses (NEMP). Perturbations to computer-based control systems that can be caused by electromagnetic disturbances are functional error modes that involve no component damage. These error modes are collectively known as "upset," can occur simultaneously in all of the channels of a redundant control system, and are software dependent. To date, there are not comprehensive guidelines or criteria for detecting upset, designing upset recovery mechanisms, or performing upset tests or analyses on digital control systems.

An upset test methodology is under development for fault-tolerant control systems. The laboratory test method involves perturbing the controller under test with transient signals that are representative of induced electromagnetic disturbances. The controller is interfaced to a simulation of the plant so that the closed-loop dynamics of the system are represented. During tests, the controller is monitored for system-level errors such as incorrect control law calculations and input/output errors as well as microprocessor errors such as invalid memory access and illegal opcode execution. In addition, the operation of the plant is monitored during tests so that cases can be flagged in which acceptable control is not maintained by the faulted controller. Issues for consideration in the development of the methodology include upset detection, the generation and coupling of analog signals that are representative of electromagnetic disturbances to a control system under test, analog data acquisition, and digital data acquisition from fault-tolerant systems. In addition to developing the methodology itself, other goals of the research are to develop on-line upset detection and correction strategies, upset susceptibility measure and margins, and an upset reliability estimation procedure.

The upset test methodology under development will be demonstrated using an Electronic Engine Control (EEC) unit as an experimental test-bed. The EEC is a control unit manufactured by the Hamilton Standard Division of United Technologies, which provides electronic controls for Pratt & Whitney aircraft engines. It is a full-authority controller and is a dual-channel system which operates with a primary/secondary channel redundancy strategy. An engine simulation to provide closed-loop dynamic with the EEC has not been obtained to date, but is included in future plans. Initial tests have been performed on the EEC to characterize nominal open-loop performance. Multiple bursts of a damped sinusoidal waveform have been

coupled into the EEC on sensor input lines. Although a comprehensive assessment of the test results is incomplete, initial results indicate that fault tolerance mechanisms in the EEC have been exercised as a result of the induced transient signals.

3.10 Spaceborne Computing

Laureen Brickley
IBM

The long-term survivability of spaceborne computers based on VHSIC/VLSI technologies and advanced computer architectures creates many challenges. A significant amount of modeling and testing of specific failure modes and error mechanisms has been accomplished, but the models we have evaluated are not sufficient for accurate prediction of long-term survivability under operational conditions. Very detailed failure mode effects analysis needs to be performed for the Central Processing Unit (CPU), memory, Input/Output (I/O), support functions, and power system. The statistical significance of each failure mode needs to be quantified, and the hardware and software fault tolerance mechanisms must be designed to contain and recover from all significant failures.

The sensitivity of space systems to transient errors or single events upsets (SEU) is important as systems become more autonomous and is therefore a significant contributor in the survivability model. The composite SEU rate depends on the radiation environment, the sensitivity of storage elements and the effective level of shielding provided to each component in the sytem. A technique has been developed to provide a detailed analysis of the single event error rate for a given system architecture. This analysis combined with out fault tolerance design approach is the basis for quantifying the survivability of a spaceborne system.

Under IR&D, Laureen Brickley is attempting to quantify the survivability of space systems. Our approach to date has been to evaluate the single event error rate at the system level. This involves evaluating the given environement for the application, the types and quantities of components, the sensitivities of each component, and the level of shielding. We have, from a customer's definition of a fail or incident, quantified the incident rate due to hard and soft failures for a given architecture. More in-depth study and expertise is required in shielding effects (geometry *vs.* spot shielding) to optimize the protection while minimizing weight. Laureen plans on investigating various models which might help her research.

She also plans on pursuing involvement with universities that are developing or using models to quantify the survivability or viability of space systems. The approach is to come up with a cost effective, reliable system approach to space systems and obtaining a method to verify the approach.

She could present at the workshop the evalution of "incident" rate for a spaceborne computer for hard and soft faults. An incident can result in an erroneous write to memory, stopping the execution of operational software or a machine error interrupt (based on a customer's definition of an incident). The evaluation of the single event error rate for the system might be of interest to others at the workshop.

3.11 Analysis and Prediction of System Reliability

Gary L. Engel
UNISYS Corporation

I am responsible for providing technical leadership in the development, application and interpretation of reliability models (MIL-217, Fault Tree and Markov) and statistical analysis tools (Poisson and Weibull pdf's) to the prediction and monitoring of system RAS performance. I also provide counsel to the UNISYS organizations, UNISYS vendors and UNISYS customers on difficult RAS issues. This includes the specification of field data collection and analysis systems. I often perform or participate in the RAS analysis of installed computer systems. I have developed a suite of PC (personal computer) based hierarchical reliablity models and statistical tools. These models and tools are based on *Lotus* 1-2-3 spreadsheet templates.

3.12 Trace Driven Evaluation of Fault-Tolerant Parallel Architectures

W. Kent Fuchs
University of Illinois

This talk will describe the collection and use of parallel computer address traces for evaluation of fault-tolerant multiprocessor systems. Recent results concerning collection of virtual address traces for distributed memory hypercube multicomputers and also shared memory multiprocessor systems will presented. Application of these address traces to evaluating recovery techniques will be described.

3.13 Modeling Recovery Time Distributions in Ultra-Reliable Fault-Tolerant Systems

Robert Geist
Clemson University

The accuracy of reliability prediction models depends critically upon the representation of fault recovery procedures, and hence upon the distribution of system recovery times. A technique for fitting distributions to empirical recovery time data that focuses on those components that dominate system reliability prediction is proposed. The technique uses Goldfarb's conjugate gradient descent search to minimize the L^2 norm of the error projected in the Laplace transform domain. A new parametric family of distributions is also suggested and is seen to provide uniformly better predictions of system reliability than the standard distributions used for this purpose, i.e. gamma, Weibull, and lognormal. Applications to several sets of real recovery time data are provided.

3.14 Conservative Reliability Predictions for Real-Time Software

Robert Geist
Clemson University

A method for obtaining time-dependent numerical estimates of the reliability of real-time software is proposed. An extended stochastic Petri net is used to represent the synchronization structure of N versions of the software, where dependencies among versions are modeled through correlated sampling of module execution times. The distributions of execution times are derived through an automated generation of test cases that is based on mutation analysis. This allows for results from software testing to be directly incorporated into the reliability measure. Since these test cases are designed to reveal software errors, including timing constraint violations, the associated execution times provide a worst case scenario. Applications to NASA's planetary lander control software are provided.

3.15 Real Time Parallel Processing Systems for High-Reliability Military Applications

Robert L. Goan
SCI Technology, Inc.

A major concern in the design and development of real-time processing systems for survivable military applications is the evaluation of fault tolerance design features. This is particularly difficult in parallel processing systems in which not only hardware but software faults can potentially propagate throughout the system if not controlled and isolated with levels of fault containment boundaries. The sophistication of Very High Speed Integrated Circuits (VHSIC) used in these advanced space systems precludes a reliance on more classical fault detection methods. Rather newer methods which enable real-time detection and isolation must be developed and verified to allow mission critical systems to operate in stressing scenarios and environments.

These methods require an in-depth method of models and simulations, live fault injection, and real-time measurement to ascertain their ability to successfully detect and isolate faults in a complex system. Critical technologies must also be evaluated for technology-driven fault types which may go undetected in laboratory fault injection tests.

Several innovative design techniques have been developed and implemented by SCI Technology, Inc., for specific use in strategic systems, to provide real-time fault detection and isolation for embedded parallel processing systems. These have been complemented by a fault tolerant operating system design which allows for the management of the system during a fault condition through proper recovery.

In addition, SCI is currently developing a high fidelity system model which not only allows the simulation of the system's performance, but which will allow the simulated injection of system faults under various system operating conditions.

SCI's main interest therefore lies in the development, modeling, implementation, and evaluation of real-time parallel processing systems for high-reliability military applications. This includes the development of advanced tools and techniques for evaluating the design.

3.16 Simulating Markovian Models of Dependable Systems

Ambuj Goyal
IBM

We present a unified framework for simulating Markovian models of highly dependable systems. Since the failure event is a rare event, the estimation of system dependability measures using standard simulation requires very long simulation runs. We show that a variance reduction technique called Importance Sampling, can be used to speed up the simulation by many orders of magnitude over standard simulation. This technique can be combined very effectively with regenerative simulation to estimate measures, such as steady-state availability and mean time to failure. Moreover, it can be combined with conditional Monte Carlo methods to quickly estimate transient measures, such as reliability, expected interval availability, and the distribution of interval availability. We show the effectiveness of these methods by using them to simulate large dependability models. We also discuss how these methods can be implemented in a software package to compute both transient and steady-state measures simultaneously from the same sample run.

3.17 Operational Failures in SPC Systems

Bjarne E. Helvik
Dr. techn. ELAB-RUNIT and
Division of Computer Systems and Telematics
The Norwegian Institute of Technology
The University of Trondheim
Norway

Analysis of operational failure data from SPC systems since 1983 has shown us that the failure pattern, *etc.* in a system deviates from what is commonly assumed in dependability modelling, for instance:

- failures occur in bursts, *i.e.*, it is no Poisson process;
- propagation of errors between physical "separate" units. Very much effort has been directed towards a mathematically precise analysis of computer system models and, in my opinion, a too small effort toward establishing models which reflect the true behaviour of computing systems.

This opinion is reflected in our current activities, where we have concentrated on the effects of software faults on the dependability of distributed systems, *i.e.*:

1. Error propagation experiment.
   The objectives are a better understanding of error propagation in a large distributed system as a basis for modelling, input to system level models and validation of system error handling capabilities.

   - Investigation of the types of errors due to software faults found in mature operational systems. (Finished)
   - Planning of an error-seeding and -propagation experiment; contact with potential partner. (On-going)

2. Error propagation modelling.
   The objective is to be able to predict the error propagation from one process (or one processing unit) to another. A preliminary model is established. (Further work awaits outcome of the experiment.)

3. System level modelling and tools.

   - System level model to account for above-mentioned effects. (Finished)
   - Use of transient/dynamic dependability measure in practical system assessment. (Further work awaits outcome of sub-activity below.)
   - Development of a modelling methodology and a tool for evaluation of systems with error propagation and other types of dependencies between subsystems. (On-going)

4. Design of systems robust to error propagation. (Currently no funding.)

3.18 Fault Injection System for the Study of Transient Fault Effects

Johan Karlsson
Chalmers University of Technology
Sweden

I have participated in the development of a system named FIST (Fault Injection System for study of Transient fault effects), which is aimed at experimental evaluation of dependable computing system. In this project, different methods for injection of transient faults into ICs have been studied, including the use of heavy-ion radiation, and power supply disturbances. The FIST system also facilitates fault injection in software simulation models of ICs. Several error detection mechanisms have been evaluated experimentally by the use of these fault injection methods. My work has involved, among other things, estimation of coverage and latency. Another way to use a fault injection system is to experimentally verify or disprove analytical models of a dependable computing system. So far, we have not done any work in this area at Chalmers, but it is an area of research I am personally interested in. Thus, as the theme of the workshop is almost exactly in line with my current work and interest I would be grateful if you would grant me the opportunity to participate.

3.19 Experimental Validation of Two Fundamental Design Techniques for Fault-Tolerant Distributed Computing

K.H. (Kane) Kim
B.J. Min
W.J. Guan
University of California–Irvine

In the UCI DREAM Lab, testbed-based evaluation of promising schemes for system-level real-time fault tolerance (tolerance of both hardware and software faults without missing the processing deadlines) has been one of the main activities for several years. Testbed-based evaluation of fault-tolerant distributed

computing schemes is based on experimental incorporation of the schemes into real-time computer network testbeds equipped with high-fidelity simulators of nontrivial real-time applications. The main objectives of our experiments have been as follows:

1. To study how the operational principles of promising but abstract fault tolerance schemes are translated into practical distributed fault detection and recovery logic effective in various real-time applications;
2. To validate implementation techniques and derive efficient OS primitives and protocols, and;
3. To measure and collect data to determine the execution overhead and logical complexity of various distributed fault tolerance schemes.

Recently, we conducted two different experimental validations by use of a testbed established. The testbed used was built around a tightly coupled network, called the Crossbar Multi-microcomputer System (CMS). It consists of 7 Zilog Z8001-based single-board microcomputers and shared memories. The shared memory modules are connected to the microcomputers via a crossbar interconnection network. A CMS kernel has been developed to support distributed cooperating application processes running on separate nodes. It provides system initialization, process synchronization mechanisms, message communication facilities, and interrupt handling, as well as special functions such as a global real-time clock and inter-microcomputer interrupt facilities. In order to obtain highly accurate validation results, a high fidelity simulation of a real-time application was developed to run on the CMS. The simulator provides generic on-board processor functions as well as a dynamic model of the application environment (sky) and sensor and actuator devices.

One experiment conducted dealt with the Distributed Recovery Block (DRB) scheme which is essentially an active redundancy scheme where multiple processors concurrently execute multiple versions of a software component and the same acceptance test. Past experiments involved applications of the DRB scheme to only one computing station. In the recent experiment, the DRB scheme was incorporated into two adjacent computing stations in order to further study implementation techniques and to obtain more accurate performance data as well as additional evidences of the real-time forward recovery capability of the scheme. The performance of the DRB scheme in terms of overhead and recovery time was measured. New understanding of efficient techniques for implementation of the scheme has been obtained, and the results demonstrate the fast forward recovery capability and logical soundness of the DRB scheme in the system in which efficient internode connection media are used.

The other experiment dealt with a scheme for handling temporary blackouts (TB's) which are extensive external electrical and electromagnetic disturbances that disrupt normal operation of electronic components and erase the contents of registers and RAMs. Such a blackout may be caused by unreliable power sources or high energy events. In a sense, the TB handling deals with global faults of a distributed system. It is indispensible in many aerospace applications where such global faults are non-negligible. In real-time distributed systems, processes must establish their recovery points in a coordinated manner so that the system can restore itself to a consistent state. As a part of the experiment of TB handling, efficient schemes of checkpointing and forward recovery have been designed. The performance of the TB handling implemented is currently being measured.

In addition to presenting a summary of the experiment results, major parts of our experimentation approaches, including the approaches adopted for fault injection and measurement of overhead and recovery time, will be critically reviewed in the presentation. The schemes experimented with are considered to be fundamental in nature in the sense that they are applicable to a broad range of application environments. The mechanisms implemented provide reference models for use by developers of systems for real applications.

3.20 Modular and Dependable Multiprocessing Systems

Israel Koren
University of Massachusetts at Amherst

We concentrate on modular architectures of dependable multiprocessing systems. The major advantage of these systems is the ability to incorporate support for graceful degradation so that the system can function in the presence of faulty components, at a lower level of performance. Another obvious advantage of modular design is easy expandability. Modular architectures for dependable multiprocessors should also attempt to avoid having single points of failures or apply some redundancy techniques to the hardcore portion of the architecture. An example of such an architecture has been described and analyzed in [Deshmukh 90][1].

For systems like the above, the traditional measures for dependability may be insufficient. These include measures such as reliability, availability, computational availability, performability and alike. In the above systems the network interconnecting all the system components is of utmost importance and measures concerning the connectivity of this network have to be considered in addition to the more conventional dependability measures. One such measure *accessibility* is defined as follows: A component is said to be accessible if it is fault-free and is connected to at least another fault-free component. This measure has been analyzed in [2] for a very specialized type of interconnecting network.

There is a need to study connectivity measures that are appropriate in the general case of modular multiprocessor architectures, devise procedures for their calculation and then combine them with other dependability measures.

3.21 The Transformation Approach to the Modeling and Evaluation of the Reliability and Availability Growth of Systems in Operation

J.C. Laprie, C. Beounes, M. Kaaniche, K. Kanoun
LAAS du CNRS
France

When dealing with the assessment of dependability the users of computing systems are interested in obtaining figures resulting from modeling and evaluation of systems, composed of hardware and software, with respect to both physical and design faults. Faced with these user requirements, hardware-and-software evaluations are far from being current practice. In order to fill the current gaps, we have defined an approach based on the hyperexponential model developed at LAAS which enables both reliability and availability of hardware and/or software systems to be evaluated, from the knowledge of the reliability growth of their components. This model is first shown to be satisfactory when applied to real data (reliability and availability of a switching system in operation); it is then shown how its markovian interpretation enables modeling of multi-component system through the transformation of stable reliability models into models incorporating reliability growth. This presentation relates to a paper accepted for FTCS-20.

3.22 Experimental Methodology

Nancy Leveson

I am interested in experimentation and have conducted several experiments on fault tolerance and fault elimination. Experiments should be conducted to establish the validity of the assumptions underlying our models. Unfortunately, too many experiments are used merely for sales–the conclusions rarely claim anything but "success" and are often not based on statistical analysis of the data. What is not needed is more

[1] S. Deshmukh and I. Koren, "A Modular, Highly Integrated Fault-Tolerant Multiple Bus Multiprocessor System," submitted for publication.

[2] I. Koren and Z. Koren, "On Gracefully Degrading Multiprocessors with Multistage Interconnection Networks," *IEEE Trans. on Reliability, Special Issue on Reliability of Parallel and Distributed Computing Networks*, Vol. *38*, pp. 82-89, April 1989.

"experiments" that show how wonderful the researcher's new technique works. Instead, there is a need to use experimentation to determine what does not work and what needs to be improved. There is also a need to compare alternative techniques to provide information for decision-making. Again, this does not mean comparing our own techniques to some other, inherently inferior technique to try to enhance our own reputation. One of the problems with conducting experiments in software engineering is that there is little foundation to build upon. Without such foundation, it is difficult to know what variables have to be held constant and to define very specific experiments. However, exploratory experiments can be used to identify critical variables and promising hypotheses to be tested in more carefully defined experiments. Finally, there is a need to teach more about experimental design in our Ph.D. programs and to use experts in experimental design to help with our experiments. Too often, the design of the experiments and the data analysis violate basic principles and invalidate any possible conclusions that are drawn. More sophistication in basic design will allow us to increase the amount we can learn from experimentation and increase the pace of progress in our field.

3.23 Fault-Tolerance Evaluation and Design

Dale L. Lomelino
Honeywell, Inc.

Evaluation of fault tolerant computer architectures begins early in the design cycle (e.g., System Design Review or Preliminary Design Review), allowing the evaluation results to impact the design without adversely affecting the development schedule. Lacking a detailed design, evaluation of the baseline flight design consists of analysis, simulation, and experimentation; each technique bringing unique capabilities to the total effort. A Fault Tolerance Evaluation Plan (FTEP) defines the methodology; describing the flight fault tolerance techniques, and coordinating each of the evaluation efforts.

Analysis of the flight design includes reliability modeling using state-of-the-art modeling tools (e.g., HARP and SURE), formal Failure Modes Effects and Criticality Analysis (FMECA), and other mathematical analysis. Reliability modeling utilizing multiple tools avoids dependency on any individual tool's assumptions. Analysis estimates the flight design's fault tolerance performance against the specified "basic" fault set. The FMECA expands on the basic fault set, developing a "derived" fault set specifically for the flight design. Finally, analysis identifies the data from simulation and experimentation needed to validate (e.g., error generation rate, error detection latencies, recovery latencies, and coverage).

Simulation is performed at each hierarchical level for both the flight and experimental systems. Starting with architectural-level and instructional-level simulations, the analysis results are validated and the flight performance is predicted. Moreover, these high level simulations can also show the impact on mission performance of each of the fault tolerance techniques under consideration, for a variety of fault conditions. Evaluation continues with more detailed RTL-level and gate-level simulations as the detailed design progresses. Modifications to the simulation to represent the development hardware and software provide validation of the experimental results. Ultimately, simulation bridges the gap between analysis and experimentation, accurately scaling the results obtained from the experimental system to the expected performance of the flight system.

Experimentation is performed with breadboard hardware and preliminary versions of the software for the critical fault tolerance techniques identified by analysis. Experimentation provides the greatest degree of confidence that appropriate fault tolerance techniques have been selected. Correct operation of the technique is demonstrated and preliminary performance results are obtained. Accelerated life testing is performed on key parts and technologies, validating the failure rates used in the reliability modeling. Rapid prototyping allows system integration to begin; exercising the hardware-software interfaces. To facilitate testing, error injection and data collection mechanisms are incorporated into both hardware and software; minimizing the need for expensive and intrusive external fault injection equipment. These mechanisms remain in the flight design to aid qualification testing, and in-operation testing and diagnostics. Experimentation validates the simulation results, and provides data that may be scaled to the expected flight system performance.

In conclusion, fault tolerance performance is predicted by analysis. Experimental data are collected on breadboard hardware and preliminary software. Simulations scale the experimental results to yield the expected flight performance; validating the fault tolerance computer architecture.

3.24 Software Reliability Experiences at the Jet Propulsion Laboratory

Dr. Michael R. Lyu
Jet Propulsion Laboratory

Outline of Interest:

- Software reliability engineering, including defining, modeling, measuring and analyzing the reliability of software and the factors that affect it.
- Fault-tolerant software, including design methodologies, experimentations and implementations, reliability and performance measurements, and quantitative assessments and evaluations.

Position Statement:

Software reliability has been recently identified as one of the most important factors for JPL software quality, since it quantitatively measures against software failures–the most unwanted event that disqualifies the software. The impact of software reliability to system reliability is tremendous. A recent software reliability study for a JPL project has dramatically reduced the predicted mean-time-between-failure of the system (including hardware and software) from 125.8 hours (without considering software failures) to 12 minutes (by considering software failures).

A list of investigations has been conducted for measuring software reliability for JPL flight and ground systems, including: Voyager, Magellan, Galileo, Deep Space Network (DSN) and Alaska SAR. These activities have provided us with critical feasibility studies of software reliability in the JPL environment. Nevertheless, such efforts have been minimal up to date. For most projects, no quantitative software reliability requirement is imposed. The widely-used JPL failure accounting system can at best provide rough software reliability measurements. The execution times between failures need to be elaborated. Moreover, the embedded nature of the software systems makes it difficult to analyze the cause of certain failures for software reliability measurement. A more active investigation into this area is launched now and will be fruitful for ensuring mission success on on-going and future JPL projects.

3.25 Predictive Methodology

Oleg A. Panfilov
NCR Corporation

The problem of selecting a subsystem mean time between failures (MTBF) to provide a required level of system availability is an unseparable part of a system design. The configuration of the future system in a great degree depends on it. If system availability requirements can not be met by a nonredundant system, different levels of HW redundancy and SW recovery procedures from system faults have to be used to meet the design objectives. Fault tolerant systems comprised from modules of different reliability as well as diversity of such modules in large multiprocessor systems complicate the issue of dependability evaluation. To make things worse, the fault tolerant systems can not be adequately characterized by only mean time between failures as well as such parameter as system availability. In general, a system after a reconfiguration may be available while a specific application will not be available. So, for such systems we have to introduce such metric as the application availability along with a classical system availability. It is important at the early stage of system design to have a methodology to specify availability requirements for different subsystems for the range of workloads of interest. This is important since different workloads generate different service demands for different subsystems and correspondingly produce different failure rates for the same systems. There is no known completed methodology to answer these questions now although some progress is done. It is important to verify any predictive methods of dependability estimates with the measured results.

3.26 Intelligent Computational Systems for Space Missions

Ann Patterson-Hine
NASA Ames Research Center

Joanne Bechta Dugan
Duke University/Research Triangle Institute

Jerry Yan
NASA Ames Research Center

The design and operation of intelligent computational systems for Space Station Freedom and future space missions is an evolutionary process which must include integrated approaches to system modeling, simulation, and experimental verification of both performance and dependability. System models used in design must be capable of evolving as the system is more completely defined and implemented, and as model parameters are more accurately measured. The approach we are taking involves the development of a new analytical technique which combines the object-oriented representation of reliability fault trees with methods for bounded approximate solution of dynamic combinatorial models. An object-oriented modeling environment enhances the storage and retrieval of a hierarchical model structure and also enhances the ability of the solution algorithm to detect modules in that structure. A dynamic combinatorial model is a combinatorial model that can capture dynamic system behavior such as sequence dependencies and fault recovery behavior. The hierarchical approach will allow multiple solution techniques and will select the best or most appropriate solution technique for the subsystem being modeled. Model parameters will first be approximated using AXE, an integrated environment for computation model specification, multiprocessor architecture specification, simulation, and data collection. Experiments on the Advanced Architecture Testbed at Ames Research Center will later provide experimental determination of model parameters as well as measures of system performance. Integration of the knowledge gained by modeling, simulating, and operating these systems will be used in the development of intelligent fault management strategies optimizing the performance and reliability of onboard computational systems.

3.27 Practical Issues in Modeling System Availability

Andrew Reibman
AT&T Bell Laboratories

There is a wide gap between commonly used system reliability models (e.g., Markov chains) and "real" system failure and repair time data. For example, although many analyses assume deterministic repair times of a constant repair rate, system repair times are often highly variable. In this talk we discuss some of the properties of "real" system availability data. We describe some techniques and software tools for analyzing life data. We then consider how the information obtained from this type of analysis can be used in system modeling and design.

3.28 Discrete Event Hypercube Models used to Evaluate Performance Tradeoffs and Understand Actual Performance Anomalies

E. Upchurch and J. Peterson
Jet Propulsion Laboratories
California Institute of Technology

We have developed detailed discrete event models of the Mark III and Hyperswitch (adaptive routing) Communications Network (HCN). These models have allowed us to evaluate and compare the performance of the two architectures under a variety of application workloads and architecture tradeoffs. For example,

we can evaluate: fixed *vs.* adaptive routing; increases in processor speeds; effect of faults on performance; bus bandwidths; architecture scalability.

Since the Mark III is an existing machine with up to 128 nodes, we have been able to modify and validate our Mark III model against actual performance. The HCN is currently being built and HCN model validation will soon follow. Validation has proven to be critical in this modeling work and we have been able to understand some performance anomalies by comparing actual Mark III program performance with detailed model statistics under a comparable workload. For example, a problem in scalability to 128 nodes was understood for a space communications flooding algorithm by model simulation and analysis.

We have found modeling a valuable aid for evaluation of design tradeoffs in early system development stages and an equally important tool for understanding actual system performance when the chosen design is implemented.

3.29 Design of *Integrity S2* Fault-Tolerant RISC Unix Minicomputer

William J. Watson
Tandem Computers

Keep in mind that I will be able to speak about the methods we use, but not the results of our analyses or the specifics of which tools proved most useful on which products.

I am interested in discussing the design of the "Integrity S2" fault-tolerant RISC Unix minicomputer we have designed here in Austin. I will be prepared to answer questions on its architecture, including such aspects as the loosely synchronized TMR CPUs, replicated fail-fast voters and I/O system, power system, customer serviceability features, reconfiguration software, diagnostic analyzer, and operating system robustness enhancements.

I also can speak about some of the tools used by Tandem for the design and analysis of our fault-tolerant machines. These include our scan design tools, our manual and automated fault injections tools and analysis techniques, and "Hot Plug" and power cycling tests.

The topics I am interested in are primarily those involved with evaluating the dependability of a computer system, beyond simple models that treat boards as "black boxes" with constant failure and repair rates. I am also interested in tools and techniques of fault injection, from selection of nodes at which to inject faults, through tools used, to evaluation of the results of the testing.

4 Summary: On Measurement and Modeling of Computer Systems Dependability

Joanne Bechta Dugan
Duke University,
W. Kent Fuchs
Ravi K. Iyer
University of Illinois at Urbana-Champaign,
Ram Chillarege
IBM T.J. Watson Research Center
and
Daniel P. Siewiorek
Carnegie Mellon University

A workshop on *Measurement and Modeling of Computer Dependability* was held at the Aerospace Corporation Facility on May 1 and 2, 1990. The workshop was organized by Ravi Iyer (University of Illinois) and Dan Siewiorek (Carnegie Mellon University) and the technical program committee included Jean Arlat (LAAS), Gerald Masson (Johns Hopkins) and Kishor Trivedi (Duke). The workshop was attended by a select group of researchers, allied scientists and academics involved in the discipline of computer dependability measurement and modeling.

The workshop objectives were to:

- Determine the state of the art in the dual areas of measurement and modeling of computer systems dependability;
- Identify the open issues at the boundaries of the two areas;
- Explore ways in which measurement and modeling can work in synergy to affect computer system design.

Since the essence of the papers in this special issue and the essence of the workshop are both consistent with measurement and modeling of computer system dependability, this paper is a relevant complement to the special issue in summarizing the presentations of the workshop panelists and the ensuing discussions among the experts participating in the workshop.

4.1 Introduction

Computer systems that are used in life-critical applications such as flight control, nuclear power plant monitoring, and space missions as well as those used in economic-critical applications such as banking and telecommunications are designed to be tolerant of faults or errors that may occur. Dependability measures, such as reliability and availability, quantify the degree of fault tolerance in a system. As the dependability criteria become more stringent, the system designs become more complex, and it becomes difficult if not impossible to accurately determine if a system design can meet these criteria. In order to assure that a proposed system meets the design goals, analysts and designers use various tools such as logical proofs of correctness, analytical models and experimental testing. Measurements are made on operational systems to obtain data on how systems behave in a naturally occurring environment. The complexity of highly dependable systems stresses currently available techniques in analytical modeling and experimental analysis.

There are at least three different types of fault tolerant systems of interest. Commercial general purpose systems and on-line transaction processing systems are designed to achieve high levels of availability, but the latter are additionally concerned with application integrity. Mission-critical fault tolerant systems are expected to achieve ultra-high levels of reliability, and are generally not repairable while on-line. Such systems are custom designs, and usually only one version is ever built, while there are typically very many

more copies of commercial systems from which to gather data. Analysis of fault tolerant systems can be done at the hardware level, the software level or at the system level. The purpose of the analysis can be for design improvements or validation. Research areas in measurement and modeling of computer dependability fall in one or more areas defined by the intersection of the three dimensions (purpose, level of analysis, target application).

Research groups in the areas of measurement and modeling of computer systems dependability have been studying these areas, but have worked together infrequently to address common problems. On May 1 and 2, 1990, a workshop on *Measurement and Modeling of Computer Dependability* was held at the Aerospace Corporation to assess the current successes and limitations in these areas, and to facilitate interactions between the two groups. The workshop consisted of two panel sessions on the first day, each followed by smaller group discussions. The second day (a half-day session) included summary presentations of the previous day's group meetings, and discussions about bridging the gap between modeling and measurement. In this report, we summarize the workshop group discussions and panel presentations.

This report is organized as follows. First we review the state of the art in measurement and modeling of computer dependability, describing the current abilities and limitations, after which the problems associated with bridging the gap between measurement and modeling is discussed. The final section of this report summarizes the points raised concerning directions for future work.

4.2 Measuring Computer Dependability

In the discussions of measuring computer dependability, several different subareas were delineated, these being broadly classified as fault injection, analysis of field data, and simulation. Simulation is generally done during the design development, fault injection is performed on a prototype and measurements are taken in the field. The points that were raised as to the state of the art in each of these subareas, as well as the major problems remaining to be addressed will be discussed in the following paragraphs.

4.2.1 Fault Injection

Fault injection is the process of deliberately inserting erroneous data or control signals in a portion of logic. Some members of the set of possible faults are not injectable, and, in typical fault injection experiments, only a portion of injectable faults are actually considered. The complexity of current VLSI systems has caused the set of injectable faults to increase to the point of intractability. However, the experimental space can be reduced by the use of sampling techniques, and models of fault manifestations based on system design.

Data concerning whether an injected fault is detected, whether recovery is successful, and how long detection and recovery take can be gathered during a fault injection experiment. From this data, some inferences can often be drawn about the overall detectability and/or latency of faults in the system being analyzed. Fault injection techniques are applicable to software as well as hardware, but fault injection has not been used as extensively in software as it has in hardware. Typical hardware injected faults include changing values at pins and in memory. Software injections include changing operators, incrementing or decrementing variables and removing statements entirely.

Fault injection can be used as part of a validation design process for hardware and software. For example, for validation of a computer interlock subsystem for a railway control application, fault injection was used in conjunction with axiomatic, empirical and physical models (LAAS).

Software fault injection performed in the laboratory on a commercial transaction processing system enhanced understanding of large system failures by revealing potential hazards that do not affect short term availability but which can cause catastrophic failure following a change in operational state (CMU, IBM).

Fault injection experiments in software helps shorten latency. Consideration of multiple variables in software reliability models (rather than only considering time in test) results in more accurate assessment of growth patterns (George Mason).

Fault injection used in a digital avionic system demonstrated that fault latency is independent of both length and instruction mix of a program, and injections based on device, gate and pin-level models showed

significant differences in fault coverage (Illinois, NASA Langley).

The current shortcomings of the fault injection method are the large space of possible faults and insertion times, and the difficulty in injecting in a controlled manner faults that are caused by environmental conditions such as voltage or temperature fluctuations, electromagnetic effects, electrostatic discharge, and radiation. Also, it is uncertain how to relate the effects of injected faults to faults that occur naturally in use.

4.2.2 Analysis of Field Data

Data on errors in operational systems has been gathered and studied by many researchers, in order to provide practical insight into failure characteristics and to facilitate diagnosis. Field data has been used to study the effects of workload on error production, to differentiate between permanent, transient and intermittent faults, and to assist in failure diagnosis.

Collection of data on hard failures in the *Cm** system was used to determine distributional fit and compare with predicted rates from MIL 217D. Data from four timesharing systems, and experimental multiprocessor and an experimental fault tolerant system was used to study transient errors. Automated monitoring and diagnosis techniques can enhance system availability by detecting trends, making fault predictions and diagnosing failures (CMU).

Analysis of field data was used to show strong correlation between the manifestation of permanent failures and the level an type of workload prior to the failure. Several measurement-based analysis techniques aided in this analysis. including smeared averaging of the workload data, clustering of like failures, match merging of workload and failures and a joint analysis of workload and failures (Illinois, CMU).

Memory data analysis was used to study fault latency, and fault latency distributions were generated. Results showed that the mean latency of a stuck-at-0 fault was significantly longer than that of a stuck-at-1 fault, and large variations in fault latency were found for different regions of memory (Illinois).

Data concerning detected software defects can be used to parameterize a reliability growth model. Further analysis of this data can help determine possible causes of defects and feedback into the development process. The defect can be separated by type (for example functional, initialization, checking or assignment) to parameterize several different reliability growth curves (IBM).

Despite these successes, there were several problems mentioned that were associated with the collection and analysis of field data, most notable being the lack of controlled experiments. Repeatability is a problem that manifests itself in two different ways. First, there are a large number of faulted components that are diagnosed as having failed in the field, but for which no problem is found when diagnostics are run off line or in the lab. Second, even in the field, there is no control over workload or naturally controlling errors. Further, it is time and space intensive to record every state change in the system, field data records are thus inherently incomplete. It is difficult to determine the level of detail that is appropriate for data collection as no standards exist. It is more difficult to relate workload characteristics to error production in a distributed workstation environment than in a mainframe configuration.

An important issue associated with field data, which merits investigation, is the fact the systems are changing rapidly. For a given commercial system, there are many different versions/configurations currently in the field, which makes it difficult to correlate data collected at various installations. The quick pace of updates causes another related problem. By the time enough data is collected to carefully analyze a system in the field, it is frequently ready to be replaced by a new generation system. Thus, the question arises as how models based on data collected from current systems can be extended to the next generation systems.

Some researchers felt that field data has limited predictive power, in that once a failure cause is determined, it is remedied, thus rendering the data useless. Several examples were given where some external environmental problem was causing repeated failures. Once the problem was isolated, it was fixed, making the data gathered during diagnosis useless in analyzing different systems.

The final set of issues raised was the proprietary nature of commercial data. Vendor cooperation is essential obtain information about failure behavior. Frequently, they are reluctant to provide the detailed information necessary to analyze data.

4.2.3 Simulation

Simulation of systems and circuits have been used to gather a variety of data related to computer dependability. At the transistor level, simulation of physical failure mechanisms, such as electromigration and hot-carrier stress, has been used to predict the life-time reliability of small circuits. At the gate and switch level, simulation has been used to study the effectiveness of error detection and correction techniques, fault coverage for testing and diagnosis, and the performance impact of designs for fault-tolerance. Behavioral level simulation has been used to study a variety of issues related to dependable computers, including effectiveness of error propagation, error confinement and error recovery, and validation of system tolerance of specific simulated failures.

Simulation can be viewed as bridging the gap between measurements and modeling. For example, simulated fault injection through software provides a flexible alternative to physical fault injection. Measurements are obtained on the simulated experiments, however the simulation is based on a model of the implemented system.

Areas of active research in simulation of computer dependability include the accuracy, performance (tractability and speed), and accessibility of the simulation. The accuracy is dependent on both the level of simulation as well as the accuracy of the system description and stimuli input. Performance is dependent on the length of the simulation required for meaningful output data, the detail of the system description, and the size of the system to be simulated. Accessibility includes the common issues of visualization of simulated results, obtaining and manipulating standardized formats of system description, and obtaining accurate stimuli input. Research work in this area described at the workshop included simulated fault injection of circuits and computers from VHDL descriptions (Stanford), Monte Carlo simulation of rare failure events in computers (IBM), validation of specific fault tolerance features (CMU, Illinois), and hierarchical simulation for VLSI circuit reliability (Illinois).

4.2.4 Common Measurement Problems

Several problems were reported that were common to fault injection, field data analysis and simulation. The first was the difficulty associated with correlating the results of the various experimental methods. How can the fault manifestations seen in the laboratory be related to the error symptoms observed in the field?

Experimental analysis of an existing system, either by detailed simulation, field data or fault injection is basically a case study of a particular system. Methodologies for generalizing results from these case studies to develop principles that are applicable to future designs are needed so as to gain maximum benefits from current analysis.

4.3 Modeling Computer Dependability

Models that assess the dependability of a computer system are used in various phases of the design and development process. Models are used to compare design alternatives, to analyze dependability/performance tradeoffs, and to optimize designs. Models are also used to study sensitivity issues of hypothetical systems, to help devise general guidelines for facilitating design.

Research work in the area of computer dependability modeling generally falls in two related areas, both of which are interested in developing more useful and accurate models. For lack of better terms, we refer to these two areas as *Model Accessibility* and *Sophisticated Models.* By model accessibility we mean the development of tools and techniques to make accurate (calibrated) models usable, particularly by designers. The development of more sophisticated modeling techniques is needed to keep pace with the increasing complexity of dependable systems. In both areas, researchers have been concerned with improving techniques for the construction of correct models and with techniques for solving the models once they are created.

4.3.1 Model Accessibility

In order for dependability to be fully integrated into the design, compact concise models of the most important aspects of a design (early life failures, coverage, software failures, reconfiguration, repair, etc.) must be derived and analyzed. Markov and semi-Markov models are capable of modeling a large variety of behaviors, but correct and complete Markov models are difficult to construct and understand. Techniques exist for the automatic generation of Markov models from alternative descriptions such as fault trees, petri nets or other descriptive languages (Duke, Michigan). These alternative languages are generally more concise and easier to understand, and capture system behavior at a less abstract level. The use of an alternative language for the specification of Markov models simplifies the construction of a correct model, but frequently has the unfortunate effect of generating extremely large models.

Several techniques have been developed that provide approximate solution of the large models (frequently with error bounds). Methods for decomposition of the overall model into independent submodels, based on structural or temporal considerations have also been proven useful in addressing the large model problem (Draper, Duke, IBM, NASA, Michigan)

Hybrid and hierarchical techniques are useful for avoiding the construction of large models. These techniques allow the specification of the system model in terms of smaller models of subsystems, which are solved (nearly) separately and whose solution are combined to form the solution for the overall model. Such modeling techniques have been used successfully to model flight control systems, high integrity process control systems, commercial systems, and fault tolerant parallel processing systems (Duke, NASA, Draper, RTI).

The major problem to be addressed with respect to model accessibility is technology transfer. If designers do not understand or do not believe the models or if the modeling tools are too cumbersome, then they will not use them. Understanding enhances validation. Several suggestions for improving technology transfer were made, including the integration of modeling and design work efforts, offering or sponsoring workshops and short courses on individual tools and techniques. Interaction between modelers and designers may be facilitated by summer employment for graduate students and summer or sabbatical leaves for faculty at industrial labs, and extended visits to university labs by industrial personnel.

4.3.2 Sophisticated Models

Research in computer dependability modeling is geared toward developing more sophisticated techniques for analyzing increasingly complex systems. Markov models are flexible enough to capture many complex behaviors, but are limited in their ability to model general distributions. More sophisticated models such as non-homogeneous Markov models and semi-markov models partially address this shortcoming. Markov and semi-Markov reward models have been used to analyze combined models of performance and dependability.

A major need for more sophisticated modeling techniques is the analysis of interaction effects. Typically separate models are derived for such system aspects as hardware and software or performance and dependability, while the interaction between these models is frequently ignored. Including interaction effects such as the effect of hardware failures on software (and vice versa), or the effects of checkpointing and failure on performance has been done (Michigan, IBM, Duke) but only for rather small models. More complex models are needed to include these effects, and once the modeling techniques are derived, they must be made understandable to non-modelers.

Another problem associated with modeling complex interactions is that doing so often produces models for which current numerical solution techniques are inadequate. An example of such a problem is called *model stiffness*. A stiff model is one in which there are many orders of magnitude difference in parameter values. Such systems require special numerical techniques or approximations. Determining bounds on these approximations can be particularly difficult. Further complications arise when the values of the input parameters are known imprecisely; errors in input parameters may outweigh numerical errors.

4.4 Bridging the Gap

Improved interactions between measurements and models can improve both areas in several ways. Measurements can supply input parameters to models and design. Sensitivity analysis on models can suggest which parameters should be measured more accurately. Model validation requires measurements to test the accuracy of the assumptions made. Models can be used to gain an understanding of field data or fault injection results.

As an example, a measurement-based performability model was based on error and resource usage data collected on a multiprocessor system. Model development proceeded from the collection of raw data to the estimation of the expected reward, and both normal and error behavior of the system were characterized (Illinois, Duke).

A typical set of inputs to both the model and the design of a dependable computer system includes information on failure rates, coverage values, repair information and sometimes performance metrics (rewards). These inputs can come from various sources, including other models, but at some point system measurements can be used to calibrate these models. Methods for bounding the estimate of the dependability of the system, given bounds on the estimates of the input parameters have been developed (IBM, Duke).

Analysis of the sensitivity of model results to input parameters can suggest which types of measurements should be considered more closely. A classic example is the coverage parameter, for which estimates of error detectability, recoverability, and timing are needed. Models have shown that the dependability of a system is extremely sensitive to this elusive parameter, while measurements to parameterize the models are still difficult to obtain. Another example given at the workshop is that of correlated failures in redundant components. Correlation can arise from design errors or environmental interference, and has been shown to have a significant impact on dependability estimates, but such a parameter is difficult to measure.

A model can be used as an input into an experiment, in that it can be used to define which parameters are being measured, or to determine stopping conditions for gathering data. Models (and other automated tools) are necessary for gaining an understanding of field or fault injection data, or for distinguishing between intermittent and transient faults from error characteristics, for example.

A balanced approach is the goal. Models need to reflect the reality of measurements, including nonstationary processes, missing data, incomplete observability. In addition to calling for much needed measurements, such as the case of coverage, perhaps more primitive measurements can be used to calculate coverage. Determination of the key parameters that can be measured and that are important to the models can be determined only by cooperative efforts between measurers and modelers.

4.5 Directions for Future Work

In addition to the issues cited in the previous sections, the following areas of investigation were identified.

1. Investigate the use of measurement data for validating models.
2. Use of measurement-based models for diagnosis.
3. Development of measurement-based or validated models for predicting impact of technology and configuration changes.
4. Coverage measurement.
5. Generation of a national data bank.
6. Operational software reliability *vs.* the manufacture of reliable software.
7. The establishment of fault tolerance/availability benchmarks (to be used in much the same way as standard performance benchmarks).

A follow-up workshop or series of similar workshops to be held to continually assess the current state of research and to further encourage cooperation between modelers, experimentalists and designers was recommended. This cooperation is clearly a multidisciplinary effort that must be supported by various levels of infrastructure, including computer corporations, granting agencies and research groups.

5 Acknowledgments

The workshop on which this report is based was sponsored by the IEEE Computer Society. Professor Iyer's work was supported by NASA grants NAG-1-613 and NAG-1-602. Professor Siewiorek's work was supported under his ONR contract number N00014-85-K-0008. Thanks are due to The Aerospace Corporation, in particular George Gilley and Mark Joseph for hosting the workshop and making all the necessary local arrangements. The editors thank the participants and the panelists for their contributions. Special thanks are also due to Laura Forsyth and Jayne Chase Loseke for their assistance in the organization of the workshop and in the generation of this report.

6 List of Attendees

1. Professor Jacob Abraham
   University of Texas at Austin
   2201 Donley Drive, Suite 395
   Austin, TX 78758

2. Dr. Jean Arlat
   LAAS–CNRS
   7 Av. du Colonel Roche
   31077 Toulouse, Cedex
   FRANCE

3. Professor Joanne Bechta Dugan
   Computer Science Department
   Duke University
   Durham, NC 27706

4. Celeste M. Belcastro
   Mail Stop 130
   NASA Langley Research Center
   Hampton, VA 23665

5. Laureen Brickley–400/041
   IBM
   9500 Godwin Drive
   Manassas, VA 22110

6. Savio Chau–MS 198-231
   CIT–Jet Propulsion Laboratory
   4800 Oak Grove Drive
   Pasadena, CA 91109

7. Dr. Ram Chillarege
   IBM
   T.J. Watson Research Center
   P.O. Box 704
   Yorktown Heights, NY 10598

8. Jeffrey Clark
   Department of Electrical and Computer Engineering
   University of Massachusetts
   Amherst, MA 01003

9. Gary L. Engel
   UNYSIS
   P.O. Box 64962
   St. Paul, MN 55264-0942

10. Professor W. Kent Fuchs
    Coordinated Science Laboratory
    1101 West Springfield Avenue
    University of Illinois
    Urbana, IL 61801

11. Dr. George Gilley
The Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009-2957

12. Charles T. Hoskinson–MB/220
The Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009-2957

13. Dr. Mei-Chen (Sandy) Hsueh
Digital Equipment Coporation
MR02-3/5E
Marlboro, MA 01752

14. Professor Ravi Iyer
Coordinated Science Laboratory
University of Illinois
1101 West Springfield Avenue
Urbana, IL 61801

15. Mr. Mohamed Kaaniche
LAAS–CNRS–TSF
7 Av. du Colonel Roche
31077 Toulouse, Cedex
FRANCE

16. Johan Karlsson
Chalmers University of Technology
S-41296 Goteborg
Sweden

17. Professor K.M. (Kane) Kim
Computer Engineering Program
Department of Electrical Engineering
University of California
Irvine, CA 92717

18. Dr. Gary M. Koob
Office of Naval Research, Code 1133
800 North Quincy Street
Arlington, VA 22217-5000

19. Professor Israel Koren
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

20. Dr. Jaynarayan Lala
Charles Stark Draper Laboratory
555 Technology Square
Cambridge, MA 02139

21. Don Lee–M8/166
The Aerospace Corporation
2350 East El Segundo Boulevard
El Segundo, CA 90245

22. Professor Ting-Ting Y. Lin
Electrical and Computer Engineering Department
Mail Code R-007
University of California, San Diego
La Jolla, CA 92093-0407

23. Dale Lomelino
Honeywell
13350 U.S. Highway 19 So.
Clearwater, FL 34624

24. Dr. Michael R. Lyu–M/S 125-233
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

25. Professor Gerald M. Masson
Department of Computer Science
The Johns Hopkins University
3400 North Charles Street
Baltimore, MD 21218

26. Professor Edward J. McCluskey
Center for Reliable Computing
ERL 460
Stanford University
Stanford, CA 94305-4055

27. Professor Douglas Miller
George Mason University
ORAS, Site
Fairfax, VA 22030

28. Dr. Ann Patterson-Hine
NASA Ames Research Center
MS 244-4
Moffett Field, CA 94035-1000

29. John C. Peterson–MS 198-231
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

30. Dr. Andrew Reibman
AT&T Bell Laboratories
Room 2L-528
Holmdel, NJ 07733

31. Dr. Thad Regulinski
IEEE Transactions on Reliability
P.O. Box 31113
Tucson, AZ 85751-9998

32. Professor Daniel P. Siewiorek
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

33. Raphael (Rafi) Some
Raytheon Equipment Division
528 Boston Post Road
Sudbury, MA 01776

34. Professor Kang G. Shin
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109

35. Professor Kishor S. Trivedi
Department of Computer Science
240 North Building
Duke University
Durham, NC 27706

36. Dr. Edwin Upchurch–MS 198-231
CIT–Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

37. William J. Watson
Tandem Computers
14231 Tandem Boulevard
Austin, TX 78728

38. Dr. Jerry Yan
NASA Ames Research Center
MS 244-4
Moffett Field, CA 94035