Use of Combined System Dependability and Software Reliability Growth Models

Myron Hecht, Herb Hecht, and Xuegao An

SoHaR Incorporated

www.sohar.com • 8421 Wilshire, Suite 201, • Beverly Hills, CA 90211

[myron, herb, xuegao]@sohar.com

keywords: dependability modeling, reliability growth, software reliability, air traffic control, CASRE, MEADEP

Abstract

This paper describes how MEADEP, a system level dependability prediction tool, and CASRE, a software reliability growth prediction tool can be used together to predict system reliability (probability of failure in a given time interval), availability (proportion of time service is available), and performability (reward-weighted availability) for a. The system includes COTS hardware, COTS software, radar, and communication gateways. The performability metric also accounts for capacity changes as processors in a cluster fail and recover. The Littlewood Verrall and Geometric model is used to predict reliability growth from software test data This prediction is integrated into a system level Markov model that incorporates hardware failures and recoveries, redundancy, coverage failures, and capacity. The results of the combined model can be used to predict the contribution of additional testing upon availability and a variety of other figures of merit that support management decisions.

1 Introduction

This paper addresses a new approach for assessment of complex distributed real time systems used in mission critical or safety critical applications. We demonstrate the combined use of traditional system reliability assessment techniques with software reliability growth models to enable the prediction of whether such systems will meet their reliability and availability requirements, and demonstrate how such an integrated model can be used for system level tradeoffs (e.g., redundancy vs. test time). Successive generations of both system and software reliability prediction methods and tools have been developed since the early 1970s. However, these techniques assumed that the software executed in a single module or node [Schneidewind96] and are therefore not sufficient to address the needs of current complex systems. By "complex systems", we mean systems that incorporate both COTS and developmental software, COTS hardware, and Internet Wide Area Networks (WANs), all of which contribute to system downtime.

Software reliability growth models use measured time between error reports or number of error reports in a time interval. In most cases, they evaluate the reduction in failure frequency during successive developmental test intervals to estimate the software reliability at the conclusion of a time period. Examples are the Schneidewind model, the generalized exponential model, the Musa/Okumoto Logarithmic Poisson model, and the Littlewood/Verrall model [ANSI92]. The primary limitation of reliability growth models is their lack of ability to model system architectures. Because information systems commonly incorporate parallelism, redundancy, and networks, the reliability of the system cannot be quantified solely by the failure rate calculated at the software module level. For example, several studies have shown that 80 to 95 percent of software failures in real-time systems are recoverable by redundant processes [Lee93, Tang95]. In such cases, software reliability growth models do not provide meaningful answers.

System reliability models use stochastic analysis and combinatorial probabilistic techniques to predict reliability. The underlying assumption in these measurement-based approaches is that the fundamental failure mechanisms are triggered stochastically, i.e., are non-deterministic ("Heisenbugs"). The most common modeling techniques are Markov chains and reliability block diagrams. Such models have been used to evaluate operational software based on failure data collected from commercial computer operating systems for more than a decade [Hsueh87, Tang92, Lee93]. Research on estimating parameters for such models, including failure rates and restoration times of both hardware and software components, been a research topic in computer engineering for 15 years [Iyer93]. System availability modeling has been used to evaluate availability for air traffic control software systems [Tang95, Tang99, Rosin99] and most recently also to the early operational phase at multiple sites. The problem with system reliability models is that they do not account for reliability growth. Thus, they can be used to assess system dependability, but not to predict such dependability during the development and testing phases.

In this paper, we demonstrate how software and system reliability models can be integrated to provide a basis for predicting system availability and to enable business or project management decisions. Examples of questions that such modeling can answer include:

- Given a known cost of downtime and current test data, what is the economically optimal point at which to stop testing? What uptime benefit will be achieved by additional testing beyond this point?

- Based on current testing results, what is the highest system availability that is likely to be achieved?
- How much more testing will be necessary in order for the system to achieve the required availability?
- Is testing or additional redundancy a better strategy for achieving availability goals?

In this paper, we will describe the application of an integrated system/software reliability growth model for an Internet web site server subsystem. We will then demonstrate how the impact of test time against capacity and availability can be assessed. Finally, we will demonstrate that substantive economic decisions on test strategies and stopping criteria can be developed using such a model.

The system reliability is assessed in the following examples using MEADEP [SoHaR00], a graphically oriented hierarchical reliability modeling tool. The software reliability prediction tool is SMERFS [Farr93], a well known and widely accepted software application for evaluation of test data for failure rate and defect discovery rate prediction.

2 Air Traffic Control System Example

To demonstrate the principle of the combined model, we will use a simplified system configuration based on the Standard Terminal Automation Replacement System (STARS) now being developed by the Federal Aviation Administration for upgrades at large airports or complexes of airports. Figure 1 shows the overall system.

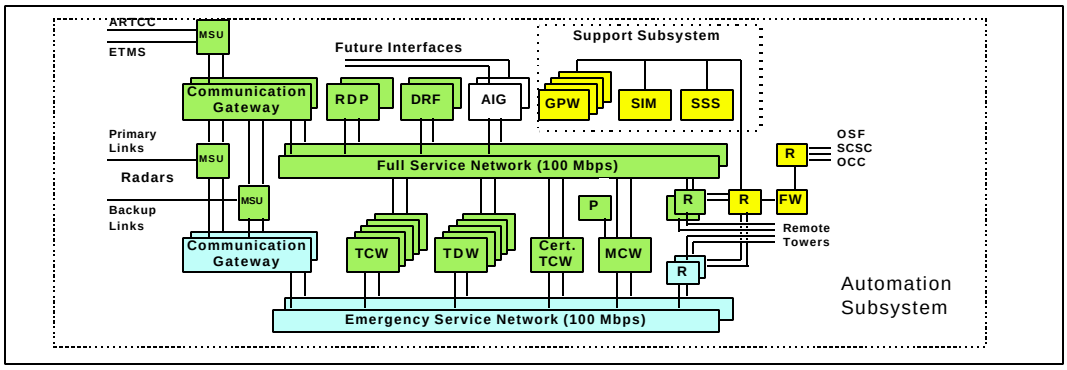


Figure 1. STARS Top Level System Diagram

Digitized radar data arrives over telephone lines and is distributed to two Communication Gateway subsystems to a primary system, designated the Full System :Level(FSL), and a backup subsystem, designated as the Emergency Service Level (ESL). Both the FSL and ESL use dual redundant switched 100 MBPS Ethernet backbones. All processors on the network use middleware provided in the Network Services (NWS) software for status reporting, registration, remote management, and related functions necessaryu for high availability. In the FSL, the data are processed by a server with redundancy running a Radar Data Processor (RDP) application that performs radar data tracking and correlation. The RDP data are then distributed to the Terminal Controller Workstations (TCWs) or larger Terminal Display Workstations (TDWs) where they are translated into a situation display. In the ESL, each workstation performs its own tracking and correlation along with the situation display; there is no central RDP server. System management is performed through other Monitor and Control Workstations (MCWs) running the System Monitor and Control (SMC) software. Other functions that support the primary mission include Data Recording and Playback (DRP) which is performed on another set of servers on the FCS network and a support subsystem performs provides site support such as simulation for test and training, adaptation edits for maps and minimum safe altitude warnings. The “P” and “R”, and “FW” designations on the diagram stand for printers, routers, and firewalls respectively.

3 Dependability Model

The dependability model for this system involves both the developed software and Commercial off the Shelf (COTS) hardware and software. Developed software undergoes testing and failure removal; its reliability growth is predicted using one of many reliability growth models, in the Computer Aided Software Reliabilty Estimation (CASRE) software package developed at JPL [Nikora94]. COTS components – whether hardware or software – have constant failure rates. We have explained above why such failures are primarily random in nature and have failure rates that can be measured using well established techniques that have been incorporated into the Measurement Based Dependability (MEADEP) .software developed by SoHaR [Hecht97].

MEADEP creates models hierarchically. A total of 11 submodels were created as shown in Figure 2. The top

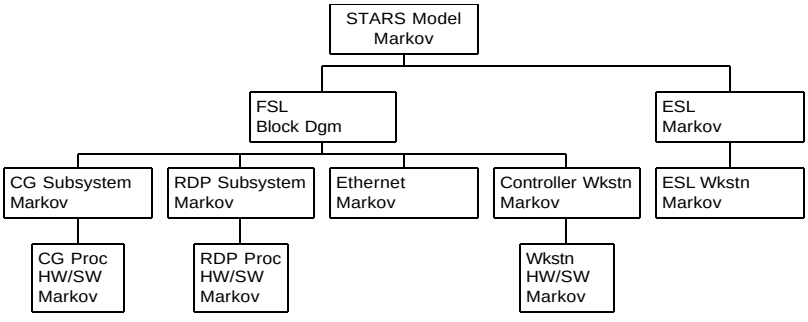


Figure 2. Model hierarchy

level diagram accounts for both the ESL and FSL integrated to provide the terminal situation display service. Within the FSL, there are submodels for each of the four major subsystems (Communications Gateway, Radar Data Processing, Network, and Controller Workstation). The ESL model is simpler, consisting only of software. With the exception of the Ethernet, which has no software, the other subsystems include both hardware, COTS software,. and developed software as will be described below. Figure 1 shows a Markov model of the radar data processing server in order to demonstrate the integration of reliability growth and stochastic reliability models. The system is available if functional in either the FSL or ESL states. It will fail if a common cause failure (power, network, massive security

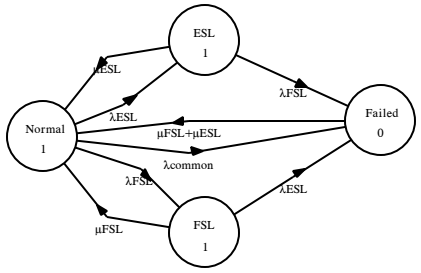


Figure 3. Top Level Markov Model

failure, etc.) brings down the entire system.

Due to length considerations, we will not consider all of the submodels. However, in order to demonstrate the integration of

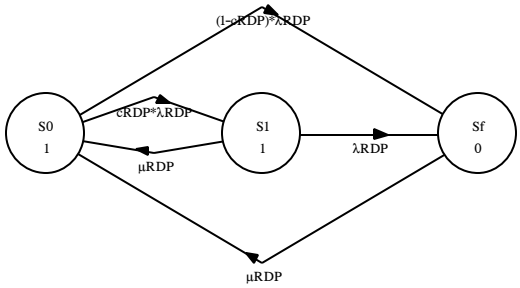
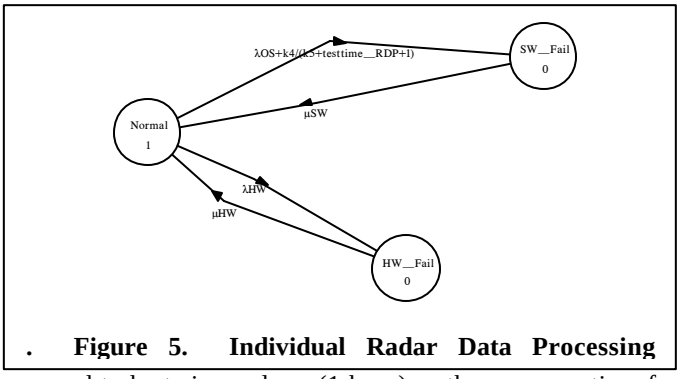


Figure 4. Radar Data Processor Submodel

reliability growth and system level modeling, we will show the Radar Data Processor subsystem (RDP) the ESL workstations.

Figure 4 shows the Radar Data Processing submodel. It is a standard Markov dual redundant model in which state S0 represents all processors up, S1 represents one processor down and the other successfully taking over, and Sf represents both processors failed. It is not certain that after a single processor failure, the second processor will successfully recover, and hence, the probably of successful detection and recovery is represented by the variable cRDP.

Figure 5 shows the detailed model of the software and hardware within the radar data processor. There are two primary failure mode categories: hardware and software. These categories are so defined because the recovery time from hardware failures is



. Figure 5. Individual Radar Data Processing

assumed to be twice as long (1 hour) as the recovery time from software failures.

The software failure rate transition is of the greatest interest in this model. Itg includes terms for the operating system failure rate and the failure rate of the radar data processing application software. The failure rate for the application software is expressed in terms of two intermediate constants, k4 and k5, which are related to the Geometric distribution [Farr96], which has the form

$$l = \frac{D\exp(b)}{Dbt_{test}\exp(b)+1} \qquad \textbf{Equation 1}$$

The constants D and β are calculated from the CASRE tool which is in turn based on algorithms developed for the SMERFS [Farr83] software. The geometric model was chosen on the basis of a best fit for the developmental data used in this example.

In Figure 5, there are three states in the model: normal,

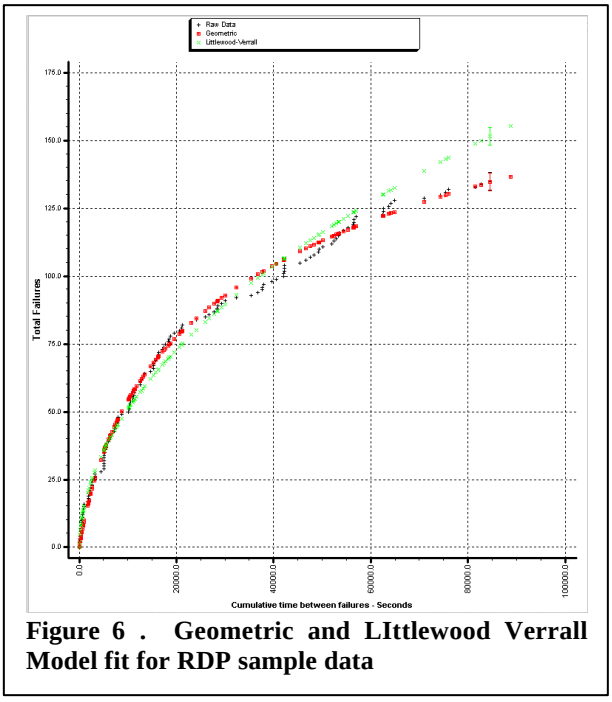


Figure 6 . Geometric and LIttlewood Verrall Model fit for RDP sample data

SW_failed (any of the major components of the RDP has suffered a failure resulting in the loss of service), and HW_failed (server hardware failure). The numbers below the state name represent rewards, i.e., the value of the state. A reward of 1

means that the system state is fully functional, a reward of 0 means that the state is completely failed. MEADEP allows intermediate values of reward (i.e., between 0 and 1) in order to represent partial loss of capability. The use of such partial rewards will be discussed below.

The following points should be noted:

- There are 4 failure modes considered in the model: hardware failure, application software failure, Internet server software failure, and operating system failure
- The hardware failure is separated from software failures because the hardware failure is restoration time requires replacement (assumed to take 1 hour) vs. a software restoration time by means of a restart (assumed to require 15 minutes), and
- The application software failure modes are assumed to be crash, hang, and stop. We have not considered the case of an incorrect response with normal behavior because it is assumed that this type of error should have been removed during earlier development. However, it could be included by the addition of a state, incorrect response could be incorporated into the system model. For this failure mode, it would be more appropriate to use a model that predicts number of faults found rather than reliability (see for example, [Farr96]).

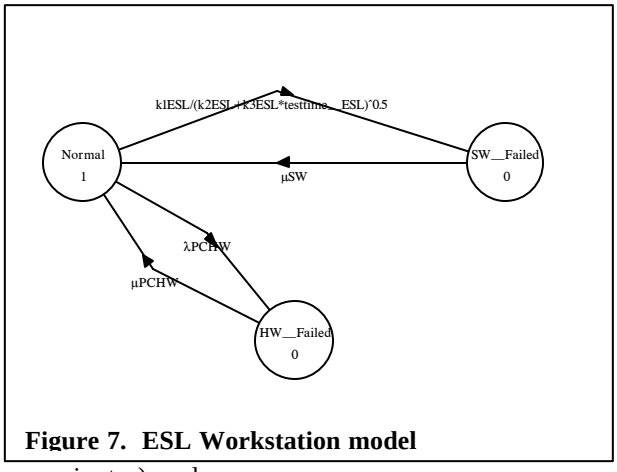


Figure 7. ESL Workstation model

For the ESL workstation subsystem, the best fit model based on the CASRE results was the Littlewood Verall Bayesian model [Littlewood80, ANSI92]. The linear of the form failure rate is [Farr96]

$$l = \frac{a-1}{\sqrt{b_o^2 + 2b_1 t_{test}(a-1)}} \qquad \textbf{Equation 2}$$

in which the values of the α, β_o, and β_1 parameters were determined by CASRE.

The ESL subsystem model differs from the FSL model because there are a number of independent workstations, each performing the entire processing for the air traffic control center independently and in parallel (this type of processing is acceptable for the reduced capabilities being provided in ESL). Figure 7 shows the ESL workstation model (lowest level in the ESL hierarchy, see Figure 1), which is similar to the RDP model in that both hardware, COTS software, and application software failures are considered, and all software failures are groupedtogether in one transition whereas all hardware failures are grouped together in the second. The constants k1, k2, and k3 represent intermediate variables of the Littlewood Verrall failure intensity relation (Equation 2) for the purposes of simplifying the model.

Figure 8 shows how the model in Figure 7 is incorporated into a higher level model that evaluates both the effects of redundancy and capacity loss..

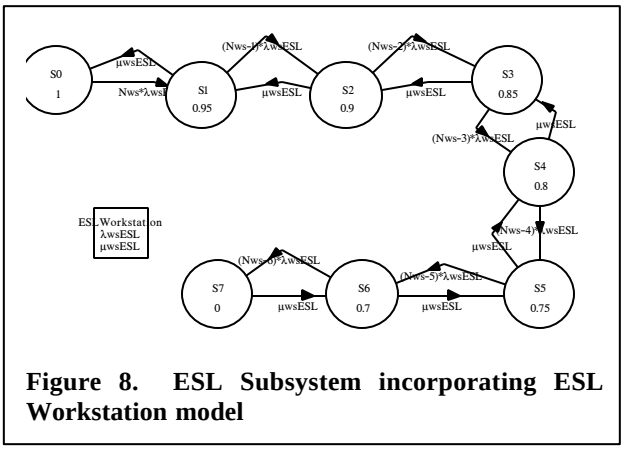


Figure 8. ESL Subsystem incorporating ESL Workstation model

Tables 1 and 2 describe the transitions and the parameters used in these models

Table 1. RDP and ESL Workstation Model Parameters

Para-meter	Explanation	Value
$\beta 1$	Parameters in Littlewood Verall linear failure intensity (see Equation 2)	1.86
α		1.21
$\beta 0$		2.54
k1	Intermediate "Dummy" variable used to simplify representation of Littlewood Verall equation in MEADEP model	α-1
k2		$\beta 0^2$
k3		$2*\beta 1*(\alpha-1)$
D	Parameters in Geometric failure intensity (see Equation 1)	.9774
β		.0105
k4	Intermediate "Dummy" variable used to simplify representation of Geometric equation MEADEP model	$D exp\beta$
k5		$\beta D exp\beta$
μHW	Repair rate of the workstation hardware, corresponding to a 1 hour MTTR	1
λHW	Failure rate of the workstation hardware, corresponding to a 1000 hour MTBF	0.0005
testtime	Test time in hours	500
λNWS	NWS middleware failure rate, corresponding to a 2000 hour MTBF	.0.0005
λOS	Failure rate of the operating system software, corresponding to a 2000 hour MTBF	0.0005

Table 2. Workstation and RDP Model Transtitions

Origin State	Desti-nation State	Expression	Comment
Normal	SW Failed	λOSk1 /(k2+k3*testtime)^.5 λOS+k4/ (k5+testtime_RDP+1)	Sum of three transition failure rates (equivalent of putting three blocks in series in a reliability block diagram): operating system, web server, and application software. The expressions k4/(k5+testtime_RDP+1) and k1/(k2+k3*testtime)^0.5 are simplifications of the Geometric and Littlewood Verall failure intensities described in Equations 1 and 2
SW Failed	Normal	μSW	Software restart time
Normal	HW Failed	λHW	Server hardware failure rate
HW Failed	Normal	μHW	Server hardware replacement/ repair time

The model consists of 7 states representing from 0 to 6 workstations failed. The rewards of each state (i.e., the numbers below the state names) represent the loss of traffic management capacity as each workstation fails (and the remaining controllers are responsible for controlling the airspace). Thus, S0, with all workstations processors functioning, has a 100% reward. The loss of a single workstation, S1, results in a 5% loss of capacity (approximately), and has the value of 0.95, the loss of two servers results in a 10% loss, and so on. Beyond 6 failures, the web site will be taken off-line because of excessive delays due to controller traffic capacity degradation.

The transitions from left to right represent failures and are all of the form

$$(Nws\text{-}n)*lws$$

in which Nws is the number of controller workstations, n is the number of failed workstations, and λws is the failure rate of each workstation (derived from evaluating the model in Figure 7). The transitions from right to left are all μws which is the restoration rate of the workstation. MEADEP defines an aggregate μ from each lower level model and hence, this restoration time represents a weighted average of the hardware and software restoration times. The frame at the bottom of the picture is how MEADEP represents that the transitions λws and μws are defined by the lower level model, PC_server, in Figure 7.

The most significant observation to be made about Figure 8 is that software reliability growth predictions have now been integrated with redundancy and with a measure of service performance. Thus, it is now possible to begin to make tradeoffs among: test time, capacity, and hardware failure rates, software failure rates, and restoration times..

4 Results

With the incorporation of the software reliability prediction results generated by CASRE, it is now possible to determine the system level reliability using MEADEP. Table 3 shows the baseline results. A complete list of the 66 parameters used in the model can be obtained from the [SoHaR00]

Table 3 . Baseline Results

Model	Failure Rate (per hour)	Recover Rate (per hour)	Availability	Unavailability
STARS	5.00E-05	3.940799	0.999987	1.27E-05
FSL	3.35E-05	1.940799	0.999983	1.72E-05
ESL	1.40E-11	2	1	7.00E-12
TCW	1.40E-09	1	1	1.40E-09
CGS	1.61E-05	2	0.999992	8.05E-06
RDPS	1.64E-05	2	0.999992	8.18E-06
Workstation	0.007105	1	0.992945	0.007054834
Gateway	0.0015	2	0.999251	0.000749438
RDP	0.001522	2	0.99924	0.000760399
ESL Workstation	0.006674	2	0.996674	0.003325875
Ethermet	1.02E-06	1	0.999999	1.02E-06

Figure 9 shows the impact of test time on workstation reliability using the Littlewood Verrall software reliability growth predictions.. Workstation reliability is an important figure of merit affecting usability of the system by controllers and air traffic safety. However, the software reliability growth models predict (in accordance with experience), that testing will have limited value beyond a certain point, and that the incremental failure rate improvement decreases with greater testing time. When the value of the additional increase in reliability improve is less than the additional testing time (however these are defined), then it is no longer cost effective to test. Moreover, alternative means of increasing reliability can also be traded off against test. For example, increasing hardware reliability or increasing redundancy might also provide the same system level result.

We now turn to the radar data processing subsystem. For this server subsystem, the primary issue is availability, i.e., the probability that the controllers will be able to receive RDP services. Figure 10 shows the results of the analysis for a single RDP processor. For the data sample used in the analysis, it is clear that relatively little benefit will accrue from additional testing. The yearly downtime will drop by some 0.45 hours per year (27 minutes) with the expenditure of approximately 900 additional hours of testing. The impact becomes even more marginal when the RDP subsystem (consisting of the primary and redundant server) is considered. In this case, the downtime drops from 0.076 hours to 0.071 hours for the same level of testing.

On the other hand, testing resources would benefit the FSL if applied at the workstation. As was the case with the ESL workstation (which uses many of the same software components including the operating system)

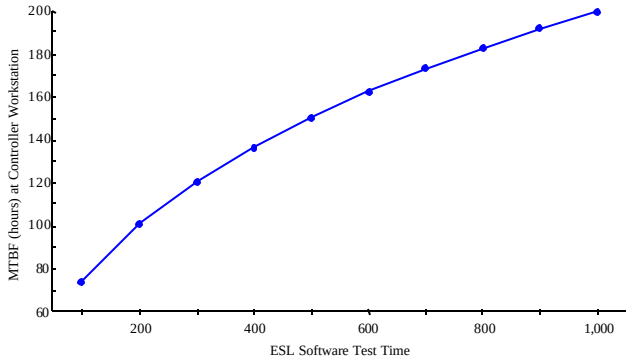


Figure 9. ESL Controller Workstation Reliability as a function of software test time

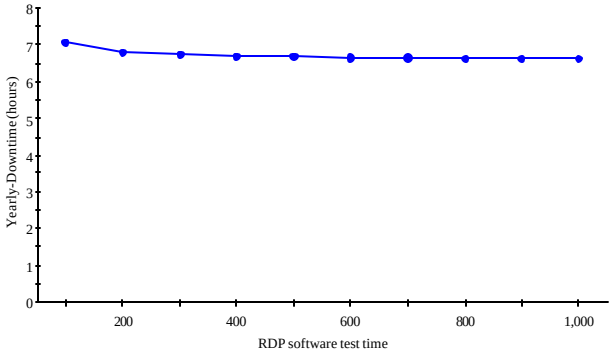


Figure 10. Impact of Software testing on downtime of a single RDP processor

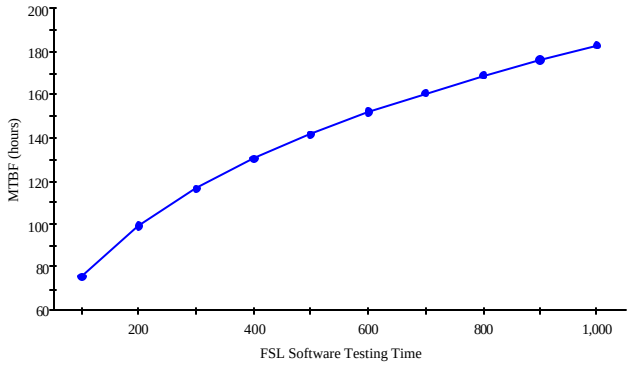


Figure 11. FSL Controller Workstation Reliability. as a Function of Testing Time

5 Tradeoffs of Software Testing vs. Other System Level Concerns

With the appropriate system model, it is possible to extend the results shown in the previous sections to the system level. For example, should resources be expended on software testing or reduction of the common mode failure rate. Figure 12 shows the impact of the common mode failure rate on the STARS system unavailability. With the baseline set of values (i.e., the system is quite reliable), the unavailability varies linearly. For the baseline assumed value of 0.00005 (corresponding to an mean time between common mode events of 20,000 hours), the unavailability is approximately 0.0000127, corresponding to a downtime of 11 minutes per year

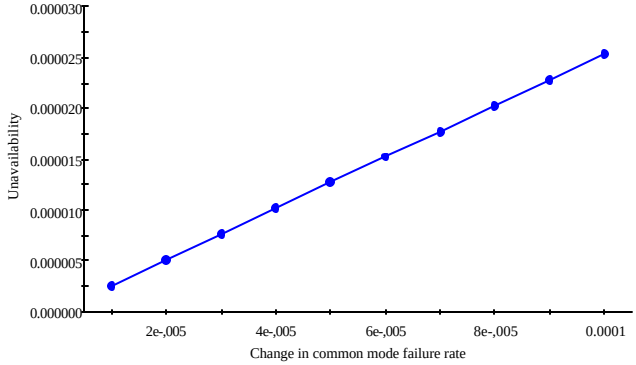


Figure 12. Impact of Common Mode Failure Rate on STARS

6 Discussion and Conclusions

This paper has demonstrated the value of integrating two mature and complementary, but until now, separated modeling and prediction approaches. As a result of this integration, it is possible to make better predictions on the economically achievable reliability of software intensive systems that incorporate redundancy and in which capacity and response time have economic value. As such the technique is also applicable to many classes of high assurance systems in space, e-commerce, military information (e.g., C^3I and logistics), and manufacturing, financial, and medical applications.

The model results should be interpreted with respect to the following limitations:

1. *Software test data*. As has been noted in many articles and papers on the subject, proper data gathering is essential for the validity of such models. The test data that are used should be gathered in accordance with the anticipated operational profile [Musa96] and should reflect a variety of testing strategies to avoid test saturation.

2. *Random failures*: The "heisenbug" assumption noted above assumes that failures occur randomly and can be modeled as a stochastic process. . This is largely true even for software failures. Deterministic software failures are easily reproduced are generally fixed early. Those which are difficult to reproduce are by definition unpredictable and therefore non-deterministic. Because most failures in software, hardware, and communication system failures are random and are the largest contributors to system downtime, the modeling and decision making approach described here is valid. However, the modeling approach does not address

deterministic phenomena such as fundamental design flaws or security challenges that have been responsible for some high visibility recent outages at large web sites.

7 References

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

[Farr93] W.H. Farr and O.D. Smith, *Statistic Modeling and Estimation of Reliability Functions for Software (SMERFS) Users Guide*, TR 84-373, Revision 3, Naval Surface Warfare Center, Dahlgren Division, September, 1993

[Farr96] W.H. Farr, "Software Reliability Modeling Survey", in *Software Reliabiltiy Engineering*, M., Lyu, ed., IEEE Computing Society Press and McGraw Hill, New York, 1996, Chapter 3

[Hecht97], M. Hecht, D. Tang, and H. Hecht "Quantitative Reliability and Availability Assessment for Critical Systems Including Software", *Proceedings of the 12th Annual Conference on Computer Assurance*, June 16-20, 1997, Gaitherburg, Maryland, USA

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

[Littlewood80] B. Littlewood, "The Littlewood-Verrall Model for Software Reliabiltiy Compared with Some Rivals", *Journal of Systems and Software*, vol. 1, no. 3, 1980

[Menn99] Joseph Menn, "Prevention of Online Crashes in No Easy Fix", *Los Angeles Times*, October 16, 1999, Section C, Page 1

[Nikora94] A. Nikora, *CASRE User's Manual*, Version 2.0, available from anikora@jpl.nasa.gov

[Rosin99], A. Rosin, M. Hecht, J. Handal, "Analysis of Airport Reliability", *Proceedings of the 1999 Annual Reliability and Maintainability Symposium*, Washington DC, USA, Jan 18-21, 1999, p. 432

[SoHaR00] MEADEP home page, www.sohar.com/meadep

[Schneidewind96], N. Schneidewind, "Software Reliability Engineering for Client Server Systems", *Proc. 7th International Symp. on Software Reliabiltiy Engineering (ISSRE)*, White Plains, New York, October, 1996, pp. 226-235

[Strauss99] Gary Strauss, "When Computers Fail", *USA Today*, page 1A, December 7, 1999

[Tang95] D. Tang and M. Hecht, "Evaluation of Software Dependability Based on Stability Test Data" *Proc. 11th Int. Symp. Fault-Tolerant Computing*, Pasadena, California, June 1995

[Tang99] D. Tang, M. Hecht, A. Rosin, J. Handal, "Experience in Using MEADEP", *Proceedings of the 1999 Annual Reliability and Maintainability Symposium*, Washington DC, USA, Jan 18-21, 1999.