

On the Efficiency of a Compound Poisson Stopping Rule for Mixed Strategy Testing

M. Sahinoglu¹, A. von Mayrhauser², A. Hajjar³, T. Chen³, Ch. Anderson²

¹ Department of Statistics
Case Western Reserve University
Cleveland, OH 44106
216-368-6013 (ph) 216-368-0252 (fax)
mesa@stat.cwru.edu

² Department of Computer Science
Colorado State University
Fort Collins, CO 80523
970-491-7016 (ph) 970-491-2466 (fax)
{anderson,avm}@cs.colostate.edu

³ Department of Electrical Engineering
Colorado State University
Fort Collins, CO 80523
970-491-6574 (ph) 970-491-2249 (fax)
{chen,amjad}@engr.colostate.edu

Abstract—When testing software, testers rarely use only one technique to generate tests that, they hope, will fulfill their testing criteria. Malaiya showed that testers switch strategies when testing yield saturates. We present and evaluate a stopping rule that can be used to determine when it is time to switch to a different testing technique, because the current one is not likely to increase criteria fulfilment. We demonstrate use of the stopping rule on a program that is being tested for branch coverage with five different testing techniques. We compare savings and accuracy of stopping both with and without using the stopping rule.

TABLE OF CONTENTS

1. INTRODUCTION
2. BACKGROUND
3. STATISTICAL METHODOLOGY
4. EXPERIMENT FOR MIXED STRATEGY TESTING
5. CONCLUSION

1. INTRODUCTION

Since testing can only show the presence of errors, not their absence, testing criteria are one way to state how much needs to be tested. The idea is that when software has been tested against given testing criteria, it has been adequately tested. Unfortunately, it is not always clear whether it will be possible to fulfill the testing criteria. For example, when testing against coverage criteria, it may not be obvious whether or not further coverage can be achieved with a given test generation technique. In practice, testers switch strategies when testing yield saturates. The question they must solve is determining the right point at which to “give up” on the current technique and switch to a new one. Beyond that is the issue of how to sequence testing techniques. Two factors will influence technique selection: cost and yield. Cost can be expressed in terms of cost of test generation (e. g. random test generation is cheaper than test generation by symbolic execution), cost of test execution (thus one wants to execute as few tests as possible and automate execution), and cost of validation (self validating tests are cheaper in this regard than tests

that require a fair amount of manual tester effort). Yield can be expressed either in coverage elements found or faults exposed. An example of coverage elements found could be the number proportion of branches covered. (alternatively As an example of faults exposed one could record failures.

We are particularly interested in developing a stopping rule for determining viability of testing techniques for increasing coverage. Section 2 summarizes possible existing approaches to this problem that range from hypothesis testing to Markov Processes. Section 3 explains the use of a Compound Poisson Process as a Bayesian Stopping Rule. We used this technique as a stopping rule, because it models the phenomenon of more than one coverage element at a time (such as with branch coverage as the criterion - a test on average covers more than one branch at a time), and includes consideration of the cost of a test with the currently applied testing strategy compared to the next one in the set of possible testing strategies. In Section 4 we evaluate the Compound Poisson approach on several thousand lines of VHDL code. The results were surprisingly strong in favor of using this stopping rule. Section 5 draws conclusions.

2. BACKGROUND

Determining when to stop testing with a given technique can be done using a variety of techniques. We include models for reliability assessment as well as coverage assessment where appropriate. In this case we replace the failure event in the model by a coverage event: “a new coverage element has just been found”¹.

Howden [5, 6] uses a simple binomial distribution to determine probability of finding another coverage element and an associated confidence interval. It is assumed that test runs are independent of each other. Testing stops when probability of finding another coverage element is low enough and associated confidence level high enough. Cost of testing is not part of the model. Other approaches similar to [5, 6] that are based on the binomial distribution include [15, 4].

¹Depending on the model, we assume this means at least one new coverage element, one new coverage element, or a specific number of coverage elements.

Various statistical testing techniques have been used to determine the number of test cases needed to achieve a particular test objective (in our case coverage related) [16, 17]. The authors apply their statistical testing formula to a variety of white box testing techniques. Stopping rules have also been used to attain a given reliability criterion [7, 20]. Poore et al. use statistical testing based on a usage model [8, 19]. This method models usage process and testing process as Markov Chains, assuming that the current state (usage as well as failure behavior) completely determines the next state. The model is more accurate than others, because it explicitly models states in the software. On the other hand it assumes memoryless behavior which may not always be the case, thus limiting the model's usability. We also have to replace failure events in the model by coverage events. A binary Markov process model [1] for random testing is able to consider relationships between consecutive test cases in terms of a correlation factor. Again, if we want to use it for coverage rather than failure modeling, we have to reinterpret failures as coverage events. Then one can use the model to determine an upper confidence bound for not obtaining any more coverage elements.

Lastly, through appropriate replacement of failure events in the models by 'new coverage' events, one could apply software reliability growth models [18]. The interpretation of failure versus "new coverage" requires some thought. If one maps, for example, each branch in the coverage model into a fault in the reliability sense, the total number of "faults" is known (i. e. the number of branches). Covering a branch in the resulting coverage model is then analogous to a failure in the reliability model. However, many models assume one failure at a time, while it is quite common to cover more than 1 branch at a time. Thus, coverage shows a certain clumping effect [9]. This limits the applicability of some of the reliability growth models and caused us to consider a Bayesian Stopping Rule that models clumping. Specifically, we considered the Compound Poisson Software Reliability Model by Sahinoglu et al., both the time-failure [9, 10, 11, 12] and the clustered (interval) failure models [13, 14].

3.1. STATISTICAL METHODOLOGY

Background and Motivation

An efficient and economical stopping rule using empirical-Bayesian principles for the Poisson counting process compounded with logarithmic series distribution(LSD) was derived and satisfactorily applied to time-domain software testing in [9, 17]. This section is a projection of this methodology to effort-based testing. The resulting compound distribution is also known as a Negative Binomial distribution(NBD) provided a certain underlying assumption (cf. equations 4 and 5 below). This empirical-Bayesian estimator is further applied to the problem of testing of software with a series of testing techniques or strategies. It is assumed that software coverage items cluster when tests are executed and that they are true-contagious in that occurrence of one coverage item adversely affects the occurrence of some others. This phenomenon is often observed in software testing practice. For example, the control structure of the program may, upon execution of some branches, prevent the execution of others. This is why the clump size distribution is assumed to be LSD while the distri-

bution of the count of testing incidents (or test-cases) is Poisson. Then, the distribution of the total number of clumped coverage items is compound Poisson [10, 11, 12, 13]. In our case, it is a Poisson LSD i.e. NBD given the assumption of equations 4 and 5.

In the analysis, we group sets of test inputs into a test suite. This grouping is likely to be problem specific (e. g. execution of a feature) or code specific (e. g. the algorithm needs to process a certain number of inputs in each step). We check for coverage increase at the end of each group of inputs. For each checkpoint in time, either the software satisfies a desired level of coverage, or software testing with the current strategy is permitted to continue. Finally, one reaches a point in time when testing with the current strategy is stopped, because coverage has saturated. Then the question is "how do we recognize the point at which we should not use the current testing strategy any longer". However, one may not be certain that all branches have been covered that the current strategy is able to cover eventually. Although there may still be a number of uncovered branches that might be executed by the current strategy, the chances of finding them within a reasonable time may be so small that it is not economically feasible to continue testing [9]. Thus, the objective is to find the optimal stopping point for the current testing strategy. Optimal stopping rule theory has been studied extensively by DeGroot in Chap.13 of his book [3]. In this section, we present an economically optimal stopping rule for the Poisson counting process compounded with the logarithmic-series distribution, based upon an economic criterion. The economic criterion trades off the cost of finding another coverage item with the current technique against the cost of switching to the next testing strategy. We are assuming that testers use a progression of techniques which become more expensive as testing progresses. Costs should include cost for test generation, execution, and validation. So, for example, a functional test may be more expensive to generate, but results might be less expensive to validate than random test generation (generation is cheap, but validation can be costly).

The number of interruptions over time or effort is distributed as Poisson, and the number of coverage items that occur as a clump at an epoch or incident is distributed according to a true-contagious logarithmic series distribution where the coverage items within a clump adversely affect and correlate with each other. A Poisson distribution compounded by a logarithmic-series will be denoted as a Poisson LSD, namely a Negative Binomial distribution (NBD) given the assumptions of equations 4 and 5.

Notation and Formulation

Given a compound Poisson process [10, 11, 12],

$$\{X(t); t \geq 0\}, X(t) = \sum_{i=1}^{N(t)} W_i \quad (1)$$

where W_1, W_2, \dots are independent, identically distributed random variables with $f(W = w) \sim \text{LSD}$ (Logarithmic series distribution)

$$f(W = w) = \frac{\alpha \theta^w}{w}, w = 1, 2, \dots \quad (2)$$

where

$$\alpha = -\frac{1}{\ln(1-\theta)} \quad (3)$$

Then, $\{X(t); t \geq 0\}$ is a Poisson/LSD process when $N(t) \sim \text{Poisson}(\lambda)$ and $w_i \sim \text{LSD}$ for $i = 1, 2, \dots$. However, if we let

$$\lambda = -k \ln(1-\theta) = k \ln q \quad (4)$$

where

$$q = \frac{1}{1-\theta} \quad (5)$$

and $k > 0$, then the Poisson/LSD process is a negative binomial distribution (NBD). Since $0 < \theta < 1$, we let the prior p.d.f. of the r.v. θ be a Beta(α, β) p.d.f. with

$$h(q) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}, 0 < \theta < 1; \alpha, \beta > 0 \quad (6)$$

and, therefore, after a series of algebraic derivations, the posterior distribution of $(\theta|x)$ is found to be as follows:

$$h(q) = \frac{\Gamma(\alpha+x)\Gamma(\beta+k)}{\Gamma(\alpha+\beta+x+k)} \theta^{\alpha+x-1} (1-\theta)^{\beta+k-1} \quad (7)$$

This is the well known Beta distribution as in Equation(6),

$$h(\theta|x) = \text{Beta}(\alpha+x, \beta+k) \quad (8)$$

With respect to the squared error loss function $L(\theta, t) = (\theta - t)^2$, the Bayes estimate of θ is given by the expected value of the posterior distribution of θ :

$$E(\theta|x) = \frac{\alpha+x}{\alpha+\beta+x+k} \quad (9)$$

We know that the expected value of the r.v. X , which is a NBD (Negative Binomial distribution), is given as follows

$$E(X) = kp \quad (10)$$

Using Equation (5) for $p = 1 - q$,

$$E(X) = k \frac{\theta}{1-\theta} \quad (11)$$

If one substitutes the Bayes estimate of r.v. θ from Equation (9) into Equation (11),

$$E(X) = k \frac{\alpha+x}{\beta+k} \quad (12)$$

Therefore $\lambda = -k \ln(1-\theta) = k \ln q$ and thus $k = \frac{\lambda}{\ln q}$ from Equation (4) can be approximated recursively by a new Equation (13) below, when the Bayes estimate of Equation (9), is entered for θ in Equation (4).

$$e^{\frac{\lambda}{k}} = \frac{\alpha+\beta+x+k}{\beta+k} \quad (13)$$

Note that in Equation (13), k can be recursively calculated after reaching convergence by applying a nonlinear solution algorithm, such as Newton-Raphson. α and β are given constants. At each step, i , we measure the following data: w_i represents the new coverage items discovered in step i . x_i is the measured cumulative coverage at step i (associated with random variable X). The associated distributions are $f(W = w) = \text{LSD}$ (Logarithmic series distribution) and $f(X) = \text{Poisson/LSD}$ or NBD when Equation (4) is assumed.

An Algorithm for a Stopping Rule

For sequential steps $i = 1, 2, 3, \dots$, i denotes the testing interval in terms of groups of test inputs applied. If the expected incremental difference between sequential steps $i = 1, 2, 3, \dots$ exceeds a given economic criterion “ d ”, we continue testing with the current strategy. Otherwise we switch to another testing strategy. Below is the one-step-ahead formula, using Equation (12) that represents this decision,

$$e(x) = E(X_{i+1}) - E(X_i) \leq d \quad (14)$$

Equation (14) can be rearranged into Equation (15) by utilizing Equation (12),

$$e(x) = k_{i+1} \frac{\alpha+x_{i+1}}{\beta+k_{i+1}} - k_i \frac{\alpha+x_i}{\beta+k_i} \leq d \quad (15)$$

where k_i are approximated at each step by Equation (13) given α and β , X_i from the input data set for each group (interval) of test inputs i is the accumulated amount of coverage. Also,

$$d = \frac{c}{a-b} \quad (16)$$

If for the i^{th} unit interval beginning at time t or for test-group i , the expected cost of stopping is greater than or equal to the expected cost of continuing, i.e., if

$$aE(X_{i+1}) \geq bE(X_i) + c \quad (17)$$

then it is economical to continue testing for the next group of test inputs. On the other hand, if the expected cost of stopping is less than the expected cost of continuing (when the inequality sign is reversed), it is more economical to stop testing with the current strategy.

$$aE(X_{i+1}) < bE(X_i) + c \quad (18)$$

If we were to stop at interval or test-group i , we assume that the cost of coverage items as yet uncovered is “ a ” per coverage item. Thus, there is an expected cost over the interval $\{i, i+1\}$ of $aE\{X_i\}$ for stopping at a discrete time $t = i$ or test-group i . If we continue testing over the interval, we assume that there is a fixed cost of “ c ” for testing, and a variable cost of “ b ” related to the uncovered coverage elements. Note that usually “ a ” is larger than “ b ” (we are assuming that subsequent testing strategies will be more expensive than the current one. This may be for two reasons: (1) increasing coverage becomes progressively harder regardless of the technique, (2) the technique itself may be more costly). Thus, the expected cost for the continuation of testing for the next time interval or test-case is $bE(X_i) + c$. This cost model is similar to that expressed in [2].

However, such estimates depend on the history of testing, which implies the use of empirical-Bayes decision procedures as described above and in the “statistician’s reward” or “secretary” problem of the optimal stopping chapter where a fixed cost “ c ” per observation is considered [3]. A cross section of stopping rules are also given in the literature by Yang [20].

Table 1: Characteristics of the Sys7 VHDL model.

Lines Of Code	3785
Branches	591
Control Bits	7
Data Bits	62
Process Blocks	62
Hierarchy	5

4. EXPERIMENT FOR MIXED STRATEGY TESTING

VHDL is a programming language that is used to express behavioral designs of hardware. The particular code we decided to experiment with is a VHDL model (Sys7) that describes an implementation of a real-time classification algorithm for two-dimensional (2D) object contours using a tree model. This is implemented in a modular very large scale integration (VLSI) architecture. The VHDL model contains 3785 lines of code and 591 branches. The system is organized with several systolic arrays as the processing elements and a global controller. The systolic array type implementation allows scalability of the architecture. However, such an implementation also increases sequential depth, thus, making testing the VHDL behavioral model more difficult.

When testing VHDL models, hardware designers commonly start with a short, limited functional test that represents common or typical usage of the design’s capabilities. This is then followed by large numbers of random inputs with varying clock cycles (clock cycles determine how long an input pattern is presented to the circuit).

The values assigned to the inputs at each clock cycle are called “test patterns”. A test pattern may be held for a number of clock cycles for reasons related to the signal depth in the design. Each test pattern may or may not cover certain branches of the code. Our goal of testing the VHDL model is to have most of the branches covered with a minimum amount of test patterns or test cases.

We subjected the VHDL code to the following four testing strategies:

1. A functional test is applied first. The functional test has 283 test patterns. These correspond to an image of a truck to be matched with three different templates.
2. 5000 random test patterns are applied. Each test pattern was held for 7 clock cycles.
3. 1000 random test patterns are applied, each test pattern was held for 4 clock cycles.
4. 5000 random test patterns are applied, each test pattern was held for 2 clock cycles.
5. 5000 random test patterns are applied, each test pattern was held for 1 clock cycle.

We generated random input patterns in two different ways: the first experiment only randomized data signals (marked DR in

the results), while the second randomized both data and control signals (marked DCR in the results).

Table 2 shows the number of test patterns applied and the cumulative branch coverage at the end of applying each test strategy for both the DR and DCR experiment.

We then applied the Compound Poisson Stopping Rule to each strategy, to see whether we could reduce the number of tests without sacrificing too much coverage. We grouped inputs into groups of 16 patterns at a time. This made sense in light of the structure of the model.

The stopping rule parameter was $d = .2E^{-6}$ with $c = 1E^{-6}$, $a = 6$, $b = 1$ (for each group of 16 input patterns). $\alpha = 8$, and $\beta = 2$. This amounts to a very conservative set-up, that is, the stopping rule is trying not to miss any branches and is likely to stop later, rather than sooner. The test coverage data represent a situation where contribution of each group of input patterns to increasing coverage is uneven, leading to a left-skewed Beta distribution. This is represented by our choice of α and β . These values model a situation where early on in the use of a new testing technique, more new coverage is more likely.

Table 3 summarizes the results for the five test strategies and the two types of randomized input generation DCR and DR when the stopping rule is applied.

Several interesting results emerged (cf. Table 3). First, the stopping rule clearly identified that the functional testing strategy should have been continued and was abandoned too quickly (the stopping rule did not recommend stopping at the end of the functional test). When going to the second strategy, the stopping rule suggested after only 64 patterns that random patterns with a clock cycle of 7 were no longer a promising strategy to increase coverage, thus saving 4,936 inputs compared to the initial strategy. Coverage with the stopping rule is lower, missing 12 branches in the DCR case and 20 in the DC case. The third strategy was considered finished after 108 inputs, saving 892 inputs. At this point, we still have less coverage using the stopping rule, we are missing 19 branches in the DCR case and 22 in the DR case. In a sense, the DR case has “caught up” a little, covering 6 new branches as opposed to 2 new ones without the stopping rule. The fourth test strategy ends with 24 fewer branches covered in the DCR case and 22 fewer branches covered in the DR case. In the last strategy, the stopping rule is suggesting stoppage without any new coverage for the DCR experiment, leaving DCR with stopping 29 branches behind. Similarly the DR strategy with stopping falls 24 branches short. In all, applying the stopping rule saved 15,180 test inputs in either experiment. Thus we

Table 2: Test Coverage Results without Stopping

Test Case	# patterns	DCR		DR	
		Branch Coverage %	Branches Covered	Branch Coverage %	Branches Covered
Functional	283	88.66	524	88.66	524
Random H=7cc	5000	93.57	553	96.45	750
Random H=4cc	1000	93.91	555	96.79	572
Random H=2cc	5000	95.26	563	96.79	572
Random H=1cc	5000	96.11	568	97.12	574

Table 3: Coverage Results using Stopping Rule

Test Case	# patterns	DCR		DR	
		Branch Coverage %	Branches Covered	Branch Coverage %	Branches Covered
Functional	283	88.66	524	88.66	524
Random H=7cc	64	90.69	536	92.05	544
Random H=4cc	108	90.69	536	93.06	550
Random H=2cc	216	91.20	539	93.06	550
Random H=1cc	432	91.20	539	93.06	550

achieved slightly lower coverage using only 6.8% of the original test cases.

While this did not happen in our experiment, the possibility of achieving higher coverage with stopping than without deserves some explanation. Consider that coverage is a function not only of the input values, but also of the internal state of the program. Thus, stopping as we did, before all inputs were executed and switching to the next strategy may find the program in a different state and cause it to execute different parts of the code for the inputs of the new test strategy than if all tests of the prior strategy had been executed. In our case, stopping was beneficial. However, the internal state of the program did not completely make up for coverage lost due to stopping.

While this result is extremely encouraging, several things must be considered before recommending this technique for large scale practical application. First, the results might depend on the particular seed used to generate the random data (although we don't expect to see large differences).

Second, how one groups the inputs for stopping rule analysis matters. We selected a group size that made sense from a model perspective. Other models will likely need different group sizes. At this point, we plan to investigate the issue of determining appropriate group sizes for the stopping rule application both model driven and coverage data driven.

Third, the choice of parameters for the stopping rule is important. The smaller d is, the more conservative the decision. On the other hand, a more aggressive stopping rule that forces stops after fewer periods of no coverage might miss branches. These results varied very little when we selected values of α and β that did not reflect the skewed data discussed earlier (i. e. $\alpha = 6, \beta = 4$).

5. CONCLUSION

This paper presented a statistical stopping rule that was used to determine when to switch testing strategies for increased code

coverage. We applied the technique to VHDL code and found that using a stopping rule yields huge improvements in testing efficiency. In the future, we would like to investigate the viability of this technique for programs other than VHDL and a variety of other testing techniques, including automated test generation. Further, how tests are grouped for the analysis is an important part of using the stopping rule. We plan to investigate ways to systematically determine a good grouping size.

Acknowledgement: This work was supported in part by the National Science Foundation through grant MIP-9628770 and a NATO-TUBITAK fellowship from Ankara, Turkey.

REFERENCES

- [1] S. Chen, S. Mills, "A binary markov process model for random testing", *IEEE Transactions on Software Engineering*, vol. 22, no. 3 (March 1996), p. 218 – 223.
- [2] S.R. Dallal and C.L. Mallows, "When Should One Stop Testing Software", *J. Am. Stat. Assn.*, 83, 872-879, 1988.
- [3] M.H. DeGroot, *Optimal Statistical Decisions*, Mc Graw Hill, New York, 1970.
- [4] D. Hamlet, J. Voas, "Faults on its sleeve: amplifying software reliability testing", *Procs. International Symposium on Software Testing and Analysis*, Cambridge, MA, August 1993, p. 89 – 98.
- [5] W. E. Howden, "Systems testing and statistical test data coverage", *Procs. COMPSAC 97*, Washington, D. C., August 1997, p. 500 – 505.
- [6] W. E. Howden, "Confidence-based reliability and statistical coverage estimation", *Procs. International Symposium on Software Reliability Engineering*, Computer Society Press, Nov. 1997, p. 283 – 291.

- [7] B. Littlewood and David Wright, "Some Conservative Stopping Rules for the Operational Testing of Safety-Critical Software", *IEEE-TSE*, Vol.23, No.11, Nov. 1997.
- [8] J. H. Poore, G. H. Walton, C. J. Trammell, "Statistical testing based on a software usage model", *Software Practice and Experience*, January 1995.
- [9] P. Randolph, M. Sahinoglu, "A Stopping Rule for a Compound Poisson Random Variable", *Applied Stochastic Models & Data Analysis*, Vol. 11, 135-143, June 1995.
- [10] M. Sahinoglu, "The Limit of Sum of Markov Bernoulli Variables in System Reliability Evaluation", *IEEE Trans. Reliability*, Vol.39, 46-50, April 1990.
- [11] M. Sahinoglu, "Negative Binomial Density of the Software Failure Count", *Proc. Fifth Int. Symp. Computer and Information Sciences(ISCIS)*, Vol. 1, pp. 231-239, October 1990.
- [12] M. Sahinoglu, "Compound Poisson Software Reliability Model", *IEEE Trans. Software Engineering*, Vol. 18, 624-630, July 1992.
- [13] M. Sahinoglu, Ü. Can, "Alternative Parameter Estimation Methods for the Compound Poisson Software Reliability Model with Clustered Failure Data", *Software Testing Verification and Reliability*, Vol. 7, No.1, pp. 35-57, March 1997.
- [14] M. Sahinoglu, A.K. Alkhalidi, (1997) "A Compound Poisson LSD Stopping Rule for Software Reliability", 5th World Meeting of ISBA, Satellite Meeting to ISI-97, Istanbul, August 1997.
- [15] A. J. van Schouwen, D. L. Parnas, S. P. Kwan, "Evaluation of Safety-critical software", *Communications of the ACM*, vol. 33, no. 6(June 1990), p. 636 – 648.
- [16] P. Thevenod-Fosse, H. Weselynck, "An investigation of statistical software testing", *Journal of Software Testing, Verification, and Reliability*, vol. 1, no. 2(July-Sept. 1991), p. 5 – 25.
- [17] P. Thevenod-Fosse, H. Weselynck, Y. Crouzet, "Software statistical testing", In: *Predictably Dependable Computing Systems*, Eds. B. Randell, J.-C. Laprie, H. Kopetz, B. Littlewood, Springer Verlag, 1995, p. 253 – 272.
- [18] M. Trachtenberg, "A general theory of software reliability modeling", *IEEE Transactions on Reliability*, vol. 39, no. 1(April1990), p. 92 – 95.
- [19] J. A. Whittaker, M. G. Thomason, "A markov chain model for statistical software testing", *IEEE Transactions on Software Engineering*, vol. 20, no. 10(October 1994), p. 812 – 824.
- [20] Mark C.K.Yang, "Comparisons of Stopping Rules and Reliability Estimation in Software Testing", SERC-TR-58-F, Purdue University, 1992.

Mehmet Sahinoglu is a visiting professor at Case Western Reserve University. Before he was head of the Department of Statistics and Dean of the College of Science and Arts at Dokuz Eylul University in Turkey. He received his PhD in Statistics from Texas A&M University and his MS in Electrical Engineering from the University of Manchester, England. He has published extensively in reliability assessment and engineering, both hardware and software.

Anneliese von Mayrhauser is a professor at Colorado State University and director of a Technology Transfer Research organization, the Colorado Advanced Software Institute. Her research interests are in software testing and software maintenance. She has published extensively in both areas. Dr. von Mayrhauser holds a PhD from Duke University and a graduate degree from University of Karlsruhe, Germany.

Amjad Hajjar received is MS in Electrical Engineering from Colorado State University. He is a PhD candidate in Electrical Engineering and is currently working as a graduate research assistant.

Tom Chen is an Associate Professor at Colorado State University. His research interests are in hardware design and testing, and hardware-software co-design. He has published extensively in this area. He received his Ph.D. in Electrical Engineering at the University of Edinburgh, U.K.

Charles Anderson is an Associate Professor at Colorado State University. His research interests are in adaptive learning and neural nets, including diverse applications in a variety of areas ranging from mechanical engineering, assistive devices and medical technology to system testing and validation. Dr. Anderson received is PhD from the University of Massachusetts, Amherst.