01 Aug 2002

Temporal Modeling of Software Test Coverage

Sahra Sedigh
Missouri University of Science and Technology, sedighs@mst.edu

Arif Ghafoor

Raymond A. Paul

Recommended Citation

S. Sedigh et al., "Temporal Modeling of Software Test Coverage," *Proceedings of the 26th Annual International Computer Software and Applications Conference (2002, Oxford, United Kingdom)*, pp. 823-828, Institute of Electrical and Electronics Engineers (IEEE), Aug 2002.
The definitive version is available at https://doi.org/10.1109/CMPSAC.2002.1045109

Temporal Modeling of Software Test Coverage

Sahra Sedigh-Ali and Arif Ghafoor
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
{sedigh,ghafoor}@ecn.purdue.edu

Raymond A. Paul
Department of Defense, OASD/C3I
Pentagon
Washington, DC 20453
ray.paul@osd.pentagon.mil

Abstract

This paper presents a temporal model for the coverage achieved by software testing. The proposed model, which is applicable at any level of the testing hierarchy, can determine the value of test coverage at any given time, as well as predicting future values. The model is comprised of two main components: coverage functions, and the coverage matrix. The coverage functions represent the coverage of a single entity as a function of time and reflect the test environment through their stochastic parameters. The coverage matrix utilizes the coverage functions to depict the coverage attained for each entity by each test within the test suite. A normalized sum of the elements of the coverage matrix is used to represent the overall coverage achieved by the test suite, as a function of time. The application of the model to multi-phase testing is illustrated. In the application section, test coverage values from Y2K compliance testing are used to verify model predictions.

1 Introduction

Software testing has been defined as the process of executing software and comparing the observed behavior to the desired behavior. The major goal of software testing is to discover errors in the software, with a secondary goal of building confidence in the proper operation of the software when testing does not discover errors [6, 5]. We define *test coverage* to be the fraction of the system that has undergone testing satisfactorily. In most cases, quantifying the extent of testing performed is a difficult, but vital task. Test coverage values and trends can be used in evaluating confidence levels of test results, or in correcting test strategies. Prediction of coverage values is even more useful, as it can be utilized in determining test durations required for achieving given coverage objectives. This paper presents a mathematical model for coverage, which can determine the quantitative value of coverage as a function of time. Section 2 provides a brief overview of past studies related to this work. Section 3 presents the coverage model and its main components. The model is generalized to multi-phase testing in Section 4. Section 5 illustrates the application of the model to test coverage data from Y2K compliance testing. Section 6 concludes our work and proposes future enhancements to the model.

This research was supported in part by the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University.

2 Related Work

Our coverage model is based on the coverage matrix introduced in [8], which represents entity-level coverage with binary elements. Our model represents coverage with a real number in the range of [0 1], facilitating the accurate representation of fractional coverage. In quantifying coverage, we represent the effects of the test environment by using two different exponential functions to determine coverage values. These exponential functions were used in [7] to model productivity. We depict two testing environments, one in which testing proceeds rapidly, and another with average testing progress. As a result, our model incorporates the notion of test duration into the model, and the elements of the coverage matrix are time-dependent. A model similar to our coverage function has been discussed in [3], with a focus on relating test coverage to known models of software reliability. The model does not incorporate the effect of test environment into coverage calculations, and does not extend the coverage analysis to the system level, as done by the coverage matrix in our model.

3 Coverage Model

3.1 Coverage Functions

In this section, we describe two exponential functions used to depict changes in coverage over time: the *rapid* and

average progress models. The former is applicable to environments where testing proceeds rapidly, and the latter to environments where testing proceeds slowly at first, then accelerates. Both models can be applied at any level and any phase of testing. The rapid progress model depicts a testing environment where testing progresses quickly, and no initial slowness is observed. This model determines $C(t)$ by the following equation:

$$C(t) = a - be^{-ct}$$

Three *test environment parameters* are involved: a, the final expected coverage, b, which can be loosely interpreted as an initial coverage indicator, and c, the testing speed indicator. All three parameters are positive real numbers. To further elaborate, a is the coverage that would be obtained if infinite time were available for testing. Coverage is represented as a fraction, hence $0 \leq a \leq 1$. Parameter b is inversely related to the initial test coverage, as $C(t) = a - b$ at time $t = 0$. If testing is performed in multiple phases, the initial coverage of the first phase of testing will be zero, hence $a = b$. In later phases, the initial coverage can be nonzero. Coverage is restricted to positive values, hence $0 \leq b \leq a$. The parameter c determines the speed of testing progress. The greater the value of c, the faster the final coverage value is achieved.

The average progress model depicts a testing environment where testing progress is initially slow, and later accelerates. This model determines $C(t)$ by the following equation:

$$C(t) = a - \frac{b}{e^{ct} + e^{-ct}}$$

The semantic meaning of the parameters is the same as in the rapid progress model. For this model, $b = 2a$ is required for initial coverage of zero, and $0 \leq b \leq 2a$ limits the coverage to positive values. Figure 1 provides a sample depiction of both progress models.

The values of the test environment parameters depend on the system under test, as well as the test method, and can be determined empirically. If all three are deterministic, the coverage can also be computed deterministically. If any parameter value is probabilistic, test coverage will be a random process. A simplified example of the analysis can be performed for the rapid progress environment, where $C(t) = a - be^{-ct}$, assuming zero initial coverage and complete final coverage, $a = b = 1$. In this case, the random process representing the coverage achieved by a given test on a given entity is determined solely by the probability distribution of the parameter c. The simplest scenario occurs when c is assumed to be uniformly distributed over [0,1]. In this case, the random process of test coverage can be represented by:

$$P(C(t) \leq x) = -ln(1-x)/t \quad \text{for} \quad 0 \leq x \leq 1 - e^{-t}$$

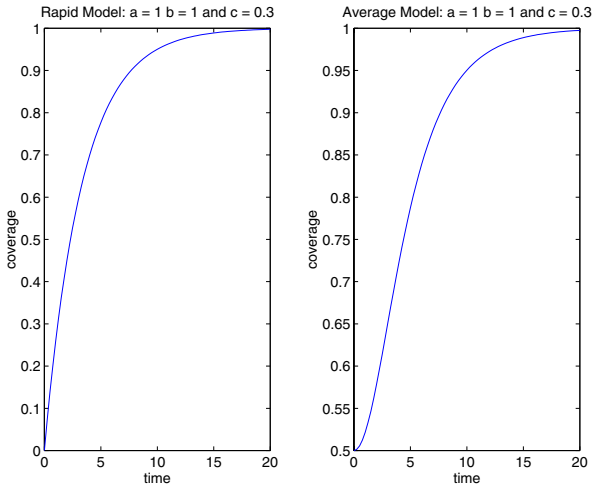

Figure 1. Rapid and average progress models.

Conversely, we may be interested in determining the probability distribution of the time T required to attain a certain coverage level, C_f. In this case, assuming the rapid progress model, $C_f = a - be^{-cT}$. Assuming that c is uniformly distributed over [0,1]:

$$P(T \leq t) = \frac{1}{t} ln \frac{b}{a - C_f} \quad \text{for} \quad t > ln \frac{b}{a - C_f}$$

As testing progresses, repairs may introduce new fault sites into the system. The modified system may have a temporary decrease in coverage until sufficient testing is performed to reach the coverage attained before repair. Additional terms should be added to the coverage function to model this transient decrease in coverage. For example, the following function, $f(t)$, can be subtracted from the coverage functions to model the decrease:

$$f(t) = \begin{cases} k(t-p)e^{-d(t-p)} & \text{for} \quad f(t) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation above, k, p, and d, respectively, determine the size, start time, and duration of the dip in coverage. Figure 2 depicts the rapid progress model, modified with the function above.

3.2 Coverage Matrix

The basic premise of our coverage model lies upon using a matrix to represent the system coverage attained by a test suite. The entities represented by the elements of the matrix can be located at any level of the testing hierarchy, from unit to end-to-end testing. The test suite can be comprised of a

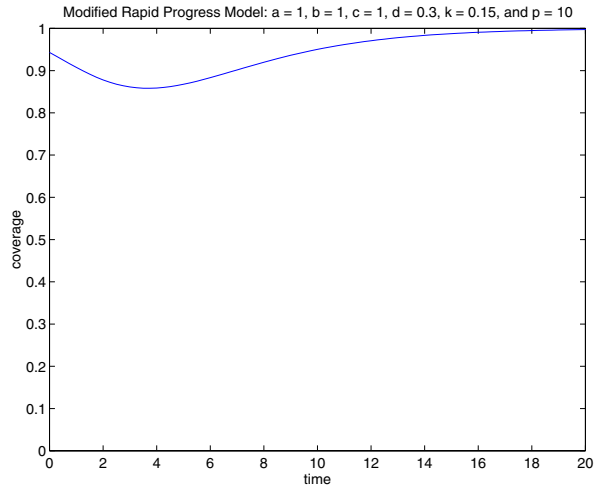

Figure 2. Modified coverage function.

single test, or a combination of any number of tests used to achieve a testing objective. In the *coverage matrix*, denoted by M, the rows represent the tests being executed, and the columns represent the entities being tested. The value of each element of M is determined by the coverage functions described in the previous section. Formally, let S be the system under test, and T the test suite applied to S. Let E be the set of entities of S that are tested by T. T and E and are assumed to be nonempty, and every element of S is assumed to belong to at least one entity in E. In the time-dependent coverage matrix $M(t)$, the rows represent elements of T, and the columns represent elements of E. Let each element $m_{ij}(t)$ of $M(t)$ be defined as the coverage obtained by running test i on entity j.

$$M(t) = \begin{bmatrix} m_{11}(t) & m_{12}(t) & \cdots & m_{1|E|}(t) \\ m_{21}(t) & m_{22}(t) & \cdots & m_{2|E|}(t) \\ \vdots & \vdots & \ddots & \vdots \\ m_{|T|1}(t) & m_{|T|2}(t) & \cdots & m_{|T||E|}(t) \end{bmatrix}$$

Depending on the test environment, $m_{ij}(t)$ are calculated by either the average or rapid progress functions. In either case, the values of the parameters, a, b, and c depend on the entity being tested, as well as the test being executed. Hence:

$$m_{ij}(t) = \begin{cases} a_{ij} - b_{ij}e^{-c_{ij}t} & \text{rapid progress} \\ a_{ij} - \dfrac{b_{ij}}{e^{c_{ij}t} + e^{-c_{ij}t}} & \text{average progress} \end{cases}$$

For mathematical tractability, we assume that the entities being tested are similar. This occurs often in module testing of software composed of similar modules performing different functions. In this case, the model parameters will depend only on the test being executed, and will be identical for all entities in the same row of the coverage matrix. Hence, all elements within the same row of $M(t)$ will be identical, as the test corresponding to row i will be expected to achieve the same coverage for all entities being tested. In this case, we represent the coverage achieved by test i, at time t, with $cvg_i(t)$. The resulting matrix will be:

$$M(t) = [\ cvg_1(t) \quad cvg_2(t) \quad \cdots \quad cvg_{|T|}(t)\ ] \times [1]_{|E|}$$

where $[1]_{|E|}$ is a vector of ones.

In the general case, where the model parameters depend on both the entity being tested and the test being executed, we define $TC_i(t)$ to be the *mean coverage* achieved by test i at time t.

$$TC_i(t) = \frac{\sum_{j=1}^{|E|} m_{ij}(t)}{|E|}$$

To obtain a measure of the *overall coverage* achieved by the test suite, we compute the mean of the values of $TC_i(t)$ over all tests in the suite. Let $CVG(t)$ be the overall coverage achieved at time t by executing test suite T for all entities in E.

$$CVG(t) = \frac{\sum_{i=1}^{|T|} TC_i(t)}{|T|} = \frac{\sum_{i=1}^{|T|}\sum_{j=1}^{|E|} m_{ij}(t)}{|T||E|}$$

The above summations rely on the implicit assumption that all entities in E undergo test i simultaneously. In the more general case, the start and end times for a test can vary among entities. The summations above will be more complex, as the coverage values corresponding to each entity e_j will be time-shifted by s_{ij}, which is the start time of test i for the entity.

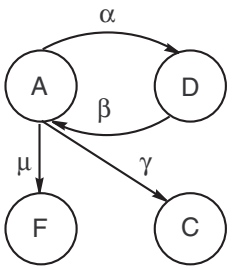

Figure 3. Entity-level state space model for testing.

In the state-space model for the general case, we can define two states for an entity that has not completed testing: *active (A)*, meaning that the entity is currently undergoing a test in the test suite, and *deferred (D)*, meaning that testing of the entity has been suspended, but will resume at some later

time. Once an entity completes testing, it reaches the *complete (C)* state, indicating that the entity has been gracefully removed from testing, either due to reaching the intended coverage goal, or due to reaching the time limit allocated for testing. An entity can also leave the test suite by *abnormal termination (F)*, meaning that testing was irrecoverably interrupted by an anomaly such as power outage or system crash. Figure 3 depicts this state-space model at the entity level. The transition probabilities α, β depend on the test schedule for the overall system, and γ can be derived from the exponential coverage function.

4 Multi-phase testing

In software regression testing, one or more test phases may be required for achieving coverage objectives. In the most general case, testing begins at Phase 1, with no initial coverage, and proceeds until the final coverage goal C_f is reached, or testing is abnormally terminated. Each test phase may have a different test environment, hence environment parameters a, b, and c will vary among phases. The state space model for multi-phase testing is illustrated in Figure 4. At test phase k, denoted by P_k, three outcomes are possible. C_f may be reached within the duration of the phase, so no further testing will be necessary. Alternatively, testing may proceed normally, but C_f may not be reached, so testing enters phase $i+1$. In the worst case, testing will be abnormally terminated. The probabilities of the aforementioned events are denoted by PC_k, PE_k, and PF_k, respectively. The number of phases, N, depends on the coverage achieved in each phase, as well as C_f. If the test environment parameters for one or more phases are non-deterministic, the total number of test phases will be a random variable.

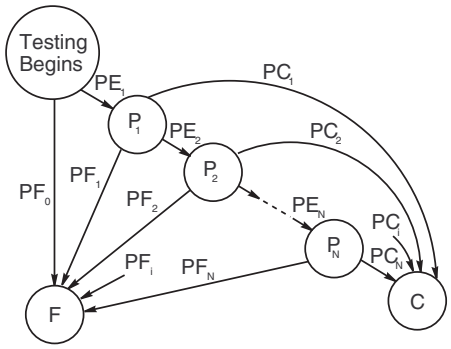

Figure 4. State space model for multi-phase testing.

Figure 5 depicts multi-phase testing with $C_f = 1$. All three test phases are assumed to have $a = b = 1$, but c = 0.2, 0.4 and 0.8 for the first, second, and third phases, respectively. This represents a typical scenario where the test environment improves over time. The phase durations are assumed to be 1, 2, and 3 time units, respectively. C_f is reached after 3.5 time units, during the third phase.

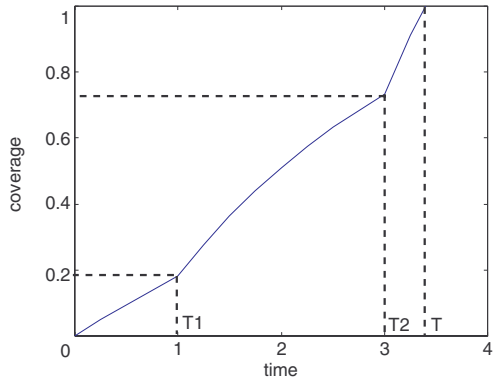

Figure 5. Multi-phase test coverage model.

5 Application

This section provides two examples of applying the model to actual test data. The data sets are from Y2K compliance testing of mission critical and non-mission critical systems. In the first data set, only one test suite is considered. In the second data set, the tests applied are classified into *replacement* and *termination* suites. For both data sets, the actual computed numbers are presented, which may exceed the limits defined for the corresponding test parameters. This is due to truncation of the time axis, corresponding to finite test time.

5.1 Y2K Compliance Testing of Mission Critical and Non-Mission Critical Systems

Figures 6 and 7 illustrate the progress made on Y2K compliance testing of mission critical and non-mission critical systems, as well as the application of the rapid and average progress models to the data. As seen in the figures, both models accurately depict the testing progress, with the average progress model yielding values that are slightly closer to the actual data. The coverage matrices follow.

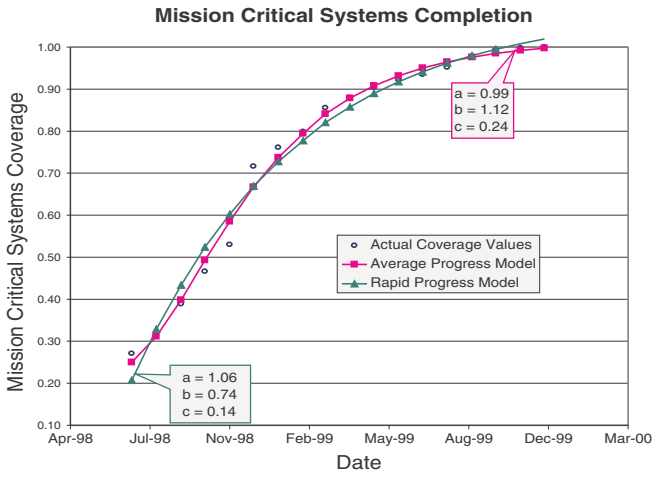

Figure 6. $m_{11}(t)$.

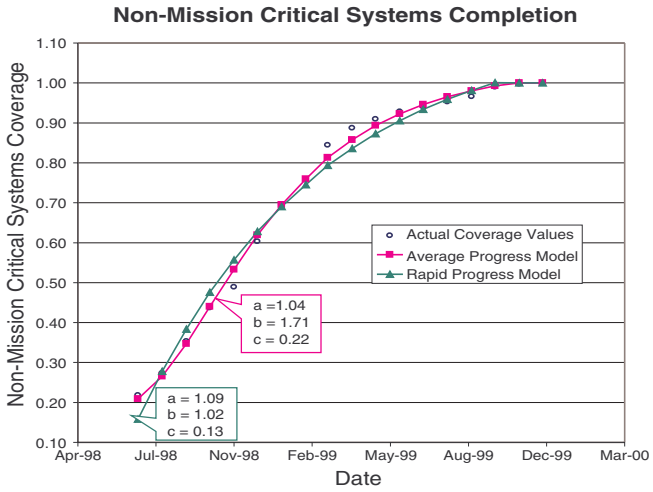

Figure 7. $m_{12}(t)$.

Rapid progress model

$$M(t) = \begin{bmatrix} 1.06 - 0.74e^{-0.14t} & 1.09 - 1.02e^{-0.13t} \end{bmatrix}$$

Average progress model

$$M(t) = \begin{bmatrix} 0.99 - \dfrac{1.12}{e^{-0.24t} + e^{0.24t}} & 1.04 - \dfrac{1.71}{e^{-0.22t} + e^{0.22t}} \end{bmatrix}$$

5.2 Y2K Compliance Testing of Mission Critical and Non-Mission Critical Systems - Repair and Termination

This section also models the progress made on Y2K compliance testing of mission critical and non-mission critical systems. Once again, the entities considered are mission critical and non-mission critical systems. In this data set, two test suites are considered, one with the objective of replacement, and another with the objective of termination of non-compliant systems. Figures 8 and 9 depict the application of the rapid and average progress models to the data. As seen in the figures, both models accurately depict the testing progress, with the average progress model yielding values that are slightly closer to the actual data. The coverage matrices follow.

Rapid progress model

$$M(t) = \begin{bmatrix} 6.33 - 5.98e^{-0.006t} & 1.92 - 1.86e^{-0.04t} \\ 1.06 - 1.73e^{-0.14t} & 1.02 - 0.65e^{-0.16t} \end{bmatrix}$$

Average progress model

$$M(t) = \begin{bmatrix} 1.19 - \dfrac{1.54}{e^{-0.11t} + e^{0.11t}} & 1.09 - \dfrac{1.85}{e^{-0.23t} + e^{0.23t}} \\ 1.06 - \dfrac{1.73}{e^{-0.14t} + e^{0.14t}} & 1 - \dfrac{1.13}{e^{-0.23t} + e^{0.23t}} \end{bmatrix}$$

6 Conclusions and Future Work

In this paper, a matrix-based exponential model was used to predict coverage values over time. Two test environments were considered, thereby incorporating the notion of "quality" of the test environment. In one model, rapid testing progress was assumed, and in the second, an initial slow period was followed by subsequent acceleration in testing progress. An entity-level state-space model was presented for both single and multi-phase testing. The model was subsequently applied to field data, and the results verified that the model could accurately represent the temporal changes in test coverage.

The temporal coverage model presented in this paper is still in the preliminary stages, and can be elaborated upon in many ways. One interesting area is the application of the model to multi-level testing by combining the results of the model at various levels of the testing hierarchy, such as unit, integration, or system testing. We plan to investigate the derivation of the components of the higher-level coverage matrix from coverage matrices at lower levels. Another interesting area of investigation is the effect of test methodology, such as model-based testing [1] or partition testing [2, 4], on the test environment parameters a, b, and c. We also intend to perform sensitivity analysis of test cover-

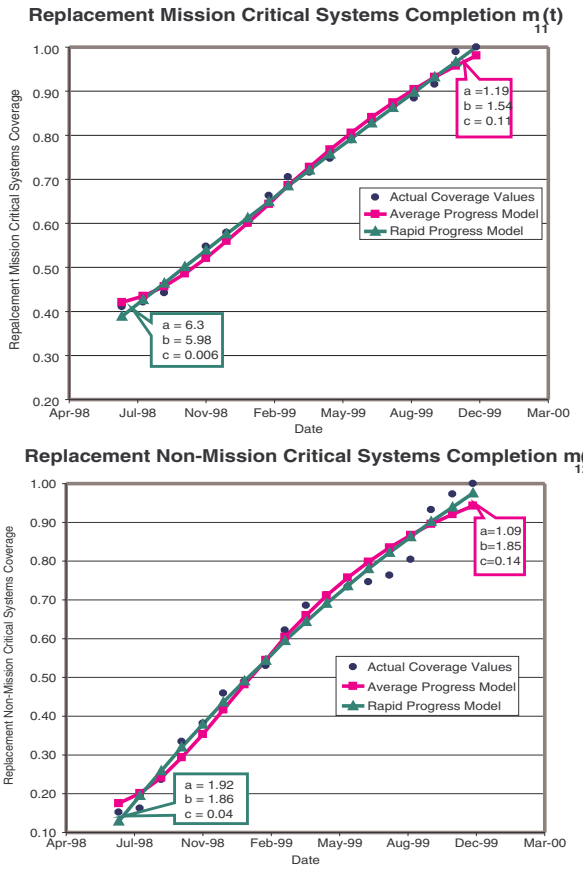

Figure 8. $m_{11}(t)$ **and** $m_{12}(t)$**.**

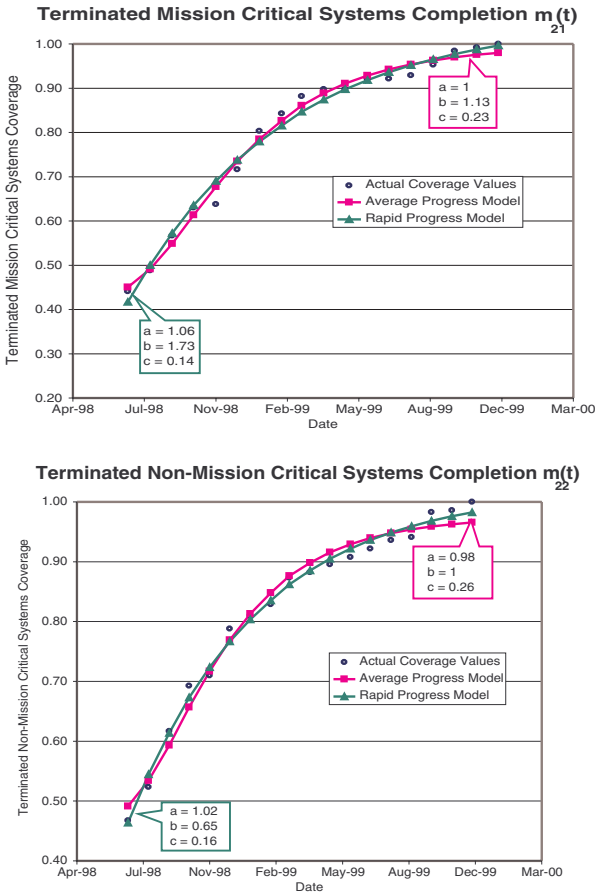

Figure 9. $m_{21}(t)$ **and** $m_{22}(t)$**.**

age with respect to the model parameters, which will enable us to investigate the relationship between test methodology and coverage.

References

[1] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz. Model-based testing in practice. In *Proc. of the 1999 Int'l Conf. on Software Engg.*, pages 285–294, May 1999.

[2] P. G. Frankl, R. G. Hamlet, B. Littlewood, and L. Strigini. Evaluating testing methods by delivered reliability. *IEEE Trans. on Software Engg.*, 24(8):586–601, Aug. 1998.

[3] S. S. Gokhale, T. Philip, P. N. Marinos, and K. S. Trivedi. Unification of finite failure non-homogeneous poisson process models through test coverage. In *Proc. of IEEE Int'l Symp. on Software Reliability Eng.*, pages 299–307, Nov. 1996.

[4] W. J. Gutjahr. Partition testing vs random testing: the influence of uncertainty. *IEEE Trans. on Software Engg.*, 25(5):661–674, Sep. 1999.

[5] D. Hamlet. Predicting dependability by testing. In *Proc. of the ACM SIGSOFT Int'l Symp. on Software Testing and Analysis*, pages 84–91, Jan. 1996.

[6] G. J. Myers. *The Art of Software Testing*. John Wiley & Sons, New York, 1979.

[7] R. A. Paul. *Software Metrics*. PhD thesis, School of Electrical Eng., Tokyo University, Tokyo, Japan, Jan. 1999.

[8] D. S. Rosenblum and E. J. Weyuker. Using coverage information to predict the cost-effectiveness of regression testing strategies. *IEEE Trans. on Software Engg.*, 23(3):146–156, Mar. 1997.