Unification of Finite Failure Non-Homogeneous Poisson Process Models through Test Coverage*

Swapna S. Gokhale, Teebu Philip, Peter N. Marinos, Kishor S. Trivedi
Center for Advanced Computing and Communication
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291
{ssg, teebu, pnm, kst}@ee.duke.edu
Phone: 919 - 660 - 5269
FAX: 919 - 660 - 5293

Abstract

A number of analytical software reliability models have been proposed for estimating the reliability growth of a software product. In this paper we present an Enhanced non-homogeneous Poisson process (ENHPP) model and show that previously reported Non-Homogeneous Poisson Process (NHPP) based models, with bounded mean value functions, are special cases of the ENHPP model. The ENHPP model differs from previous models in that it incorporates explicitly the time-varying test-coverage function in its analytical formulation, and provides for defective fault detection and test coverage during the testing and operational phases. The ENHPP model is validated using several available failure data sets.

*This work was supported in part by the US AIR FORCE Rome Laboratory as a core project in the Center for Advanced Computing and Communication and by a contract from the Charles Stark Draper Laboratory.

1 Introduction

The past several years have seen the formulation of numerous analytical models for predicting reliability and fault content of software systems. The accuracy of these models, however varies significantly[11] and no single model is known to perform well in all contexts. A major difficulty in software reliability practice is to analyze the context in which reliability measurement is to take place. As a result, there is no definitive model that can be recommended unreservedly, and potential users are left in a dilemma as to which models to choose, which procedures to apply, and which predictions to trust, among the various competing options.

Accurate predictions of software release times, and estimation of the reliability and availability of a software product require quantification of a critical element of the software testing process: **test coverage**.

In this paper, we present the Enhanced Non-Homogeneous Poisson Process model, which accounts explicitly for test coverage and show that previously reported Non-Homogeneous Poisson Process (NHPP) software reliability models, with bounded mean value functions, are special cases of the ENHPP model. The ENHPP model is capable of handling any general coverage function and is thus an important step towards the unification of the finite-failure NHPP models, which rely on specific coverage functions. The formulation allows for imperfect test coverage and imperfect detection coverage besides providing a new decomposition of the mean value function. In the next section, a unified definition of test coverage is provided.

2 Background and Motivation

Program testability and test coverage are related concepts. The first refers to the ease with which one may test a program while the second provides a measure of how thoroughly a test has "covered" all potential fault-sites in a program. A *potential fault-site* is defined very broadly to mean any structurally or functionally described program element whose integrity may require verification and validation via an appropriately designed test. The factors which impact program testability and test coverage include: (i) *program complexity*, (ii) *software development philosophy or approach*, (iii) *software tools employed*, (iv) *test quality*, and (v) *test effectiveness*.

2.1 Existing Notions of Coverage

The importance of test coverage has been recognized by several researchers [8, 10], and empirical evidence strongly suggests that testing which is carried out without some form of test coverage measurement may fail to sensitize as much as 45% of the code. Software designers and testers must develop effective evaluation tools capable of measuring test coverage and pointing out design deficiencies contributing to poor software testability. Such tools should also provide data which lead to improvements in the quality and effectiveness of a software test as well as information about its adequacy when deciding product-release times.

There are two types of coverage definitions in literature: control-flow and data-flow coverage[4, 8, 9, 16]. Each coverage criterion proposed in the literature captures some important aspect of a program's structure. In general, *test coverage* is a measure of how well a test covers all the **potential fault-sites** in a software product under test. It should be obvious that how one defines a *potential fault-site* and how well such fault-sites are sensitized influence greatly the significance of this metric. Potential fault-sites are introduced here to mean program entities representing either structural or functional program elements whose sensitization is deemed essential towards establishing the operational integrity of the software product. Specific instances of potential fault-sites range from source-language and machine-code statements to high-level specification language statements and program blocks with I/O specification and functional description, only.

The first important step is taken here towards offering a unifying definition for *test coverage* which accommodates all the proposed specializations of the concept(i.e, statement coverage, block coverage, decision coverage, condition/decision coverage, etc.) without the burden of the specificity they impose on the modeling process used to estimate or predict the reliability of software products.

Definition 2.1 : *(Test Coverage): Given a software product and its companion test set, one defines test coverage to be the ratio of the number of potential fault-sites sensitized by the test divided by the total number of potential fault-sites under consideration.*

As mentioned earlier, there are several definitions of test coverage but the one offered here is most general and easily adaptable to situations which may benefit from a specialized application of the concept. The definition allows also for the possibility of defective or imperfect test coverage; which is defined as the inability to sensitize all potential fault-sites through an infinite number of tests.

In Section 3, a framework is developed which incorporates test coverage into software reliability modeling.

3 Relating Test-Coverage to Software Reliability: The ENHPP Model

The proposed ENHPP model is based on the following assumptions:

- faults are uniformly distributed over all potential fault sites,
- when a potential fault-site is sensitized at time t, any fault present at that site is detected with probability $c_d(t)$, and
- repairs are effected instantly and without introduction of new faults.

Analytically, the model is based on the expression

$$\frac{dm(t)}{dt} = \tilde{a} c_d(t) \frac{dc(t)}{dt}. \qquad (1)$$

or

$$m(t) = \tilde{a} \int_0^t c_d(\tau) * c'(\tau) d\tau \qquad (2)$$

where $\tilde{a}$ is defined as the total number of faults which are expected to be detected given infinite testing time, perfect fault detection coverage implying $c_d(t) = 1$, and perfect test coverage leading to $c(\infty) = 1$. The expected number of faults detected by time t is $m(t)$. If $c_d(\tau)$ is assumed to be a constant value K, then, we have,

$$m(t) = \tilde{a} K c(t). \qquad (3)$$

Equation (3) is intuitively simple: it simply states that the expected number of faults detected by time t is equal to the total number of faults in the program times the probability of detecting a fault times the fraction of potential fault sites covered by time t.

Substituting $a = \tilde{a}K$, Equation (3) can be written as:

$$m(t) = ac(t) \qquad (4)$$

This results in a failure intensity function $\lambda(t)$

$$\lambda(t) = \frac{dm(t)}{dt} = ac'(t) \qquad (5)$$

The failure intensity function can also be rewritten as:

$$\lambda(t) = [a - m(t)] \frac{c'(t)}{1 - c(t)} \qquad (6)$$

Equation (6) shows clearly that failure intensity depends not only on the number of remaining faults, but, also, on the rate at which the remaining potential fault sites are sensitized (covered) divided by the current, fractional population of uncovered (unsensitized) potential fault sites.

The failure occurrence rate per fault, or the hazard function, $h(t)$, is thus given by

$$h(t) = \frac{c'(t)}{1 - c(t)} \qquad (7)$$

which accounts for product testability and test effectiveness, explicitly.

The ENHPP model allows for the fact that as testing proceeds, the rate at which an individual fault will surface can vary with time and thus permits time-dependent failure occurrence rate per fault. Furthermore, it shows that the "distribution" corresponding to this hazard function is precisely the coverage function evaluated at time t.

The ENHPP The ENHPP model can accommodate the realistic situation of a defective coverage function, that is $c(\infty) < 1$, as shown in Figure 1.

The resulting reliability expression $R(t|s)$ is as follows:

$$R(t|s) = e^{-\int_s^{s+t} \lambda(\tau) d\tau} = e^{-a[c(s+t) - c(s)]} \qquad (8)$$

where s is the time of the last failure and t is the time measured from the last failure.

4 Existing NHPP Models and Coverage Functions

The time domain models which assume the failure process to be a NHPP differ in the approach they use for determining $\lambda(t)$, and thus the coverage function $c(t)$. The test-coverage function can have several different forms, some of which are discussed below.

4.1 Exponential Coverage Function

The popular Goel-Okumoto(GO) model[5] has had a strong influence on software reliability modeling. It is a very simple model whose parameters have a physical interpretation. Specifically, the GO model states that the failure process is modeled by NHPP, with a mean value function $m(t)$ given by

$$m(t) = a(1 - e^{-gt}) \tag{9}$$

where a is the expected number of faults that would be observed by the testing process given infinite testing time, and g is the failure occurrence rate per fault.

The failure intensity function $\lambda(t)$ is given as

$$\lambda(t) = age^{-gt} \tag{10}$$

The coverage function and the failure occurrence rate per fault for the GO model are given in Equations (11) and (12), respectively.

$$c(t) = 1 - e^{-gt} \tag{11}$$

$$h(t) = \frac{c'(t)}{1 - c(t)} = g \tag{12}$$

Equation (12), shows that the failure occurrence rate per fault is constant in the case of an exponential coverage function.

4.2 Weibull Coverage Function

In the case of an exponential coverage function, the software system exhibits a decreasing failure intensity pattern during testing, i.e., the software quality continues to improve as testing progresses. However, in most practical situations, the software failure intensity increases initially and then decreases. The generalized Goel-Okumoto(GO) model[1, 2] was proposed to capture this increasing-decreasing failure intensity function. Its mean value function is given by

$$m(t) = a(1 - e^{-gt^{\gamma}}) \tag{13}$$

where a is the expected number of faults that would be observed by the testing process given infinite testing time, and g and γ are parameters reflecting the quality of testing.

The failure intensity is given by

$$\lambda(t) = ag\gamma e^{-gt^{\gamma}} t^{\gamma-1} \tag{14}$$

The coverage function and the failure occurrence rate per fault or hazard function are given by Equations (15) and (16), respectively.

$$c(t) = 1 - e^{-gt^{\gamma}} \tag{15}$$

$$h(t) = \frac{c'(t)}{1 - c(t)} = g\gamma t^{\gamma-1} \tag{16}$$

Equation (16) shows that the Weibull coverage function gives rise to a time-varying failure occurrence rate per fault. The hazard function, $h(t)$, is an increasing function of time for $\gamma > 1$, decreasing for $\gamma < 1$, and reduces to a constant for $\gamma = 1$.

4.3 S-Shaped Coverage Function

The s-shaped software reliability growth model[14] was proposed to capture the realism of is a time delay between the fault detection and fault removal. The testing process in this case can be seen as consisting of two phases: fault detection and fault isolation/removal.

The mean value function is given by

$$m(t) = a[1 - (1 + gt)e^{-gt}] \tag{17}$$

where a has the same interpretation as before, and g is the fault removal rate parameter.

The failure intensity function is given by

$$\lambda(t) = g^2 t e^{-gt} \tag{18}$$

The coverage function and the failure occurrence rate per fault are given by Equations (19) and (20), respectively.

$$c(t) = 1 - (1 + gt)e^{-gt} \tag{19}$$

$$h(t) = \frac{c'(t)}{1 - c(t)} = \frac{g^2 t}{1 + gt} \tag{20}$$

Figure 2 and 3 show typical coverage functions and the corresponding hazard functions.

5 Model Validation

5.1 Parameter Estimation

In order for a software reliability model to be of any practical value, its unknown parameters must be estimated using real failure data. Failure data are usually available in one of two forms:

- Time data : The time data consists of times between successive software failures. This is the most detailed form of data from the estimation point of view.
- Group data : The group data is reported in the form of number of failures n_i, experienced in a testing interval i, of duration t_i.

The estimation of parameters using time data is important for evaluating the predictive capability of a model. It has been discussed in the literature[5] and is used here.

5.2 Predictive Capability of the Model

Existing software reliability models can produce very different results when called upon to predict future reliability. The raw data available to the user is usually in the form of a sequence of times between successive failures, and the user is interested in using the observations of the past to predict the times of software product failure in the future. The focus of this paper is on the simplest form of prediction, i.e., use of the data to predict the current reliability of the software product under consideration, and then analyze the predictive capability of the model using u-plots[3].

5.3 Software Failure Data and Analyses

Parameter estimation and validation techniques are discussed in the literature and are used here in conjunction with various software failure data sets. The NTDS data[5] are from the U.S. Navy Fleet Computer Programming Center, and is referred to as data set-1. Analysis of three additional data sets originally compiled by Musa[13], and later referred to by Littlewood[3] are provided; those data sets are referred to as data sets 2, 3 and 4, respectively.

5.3.1 Observed and Estimated Mean Value Functions

The estimated mean value function is computed for each of the models(i.e., ENHPP and its precursors) and is plotted along with the observed mean value function for all four failure data sets.

Figure 4 gives the mean value functions for data set 1 (actual) and those estimated for the exponential and s-shaped and Weibull coverage functions.

The error sum of squares between the predicted and observed mean value functions are summarized in Table 1. It is evident that the best approximation of the observed mean value function is achieved by different coverage functions for different data sets, in a least error sum of squares sense.

5.3.2 u-plots

A u-plot provides a visual method of determining a software reliability growth model's goodness of fit. Figure 5 shows the u-plots for data set 2. The Kolmogorov distance, a numerical technique for estimating goodness of fit, is summarized in Table 2.

6 Conclusions and Future Work

In this paper we have presented an Enhanced Non-Homogeneous Poisson Process (ENHPP) software reliability model which allows the explicit incorporation of test coverage, which can be defective. Furthermore, the model allows for imperfect detection coverage and provides a new decomposition for the mean value function. We have validated the ENHPP model for three coverage functions by comparing their observed mean value functions with the estimated mean value functions obtained from actual failure data. The predictive capability of ENHPP was also assessed and the results obtained clearly indicate that the choice of coverage function depends on the criteria which best meet the user's specific needs. The utility of existing NHPP models is limited since they provide only for a single coverage function. The ENHPP model, however, allows the use of a generalized coverage function, which can be selected in a way that best meets the user's requirements. The ENHPP model is thus a key step towards unifying the NHPP models.

7 Acknowledgments

The authors wish to thank Dr. Babubhai V. Shah of Research Triangle Institute, RTP, for his technical assistance. The authors also thank the anonymous referees for their excellent suggestions.

References

[1] "A Guidebook for Software Reliability Assessment," Tech. Rep. RADC-TR-83-176, Aug. 1983.

[2] "Software Reliability Modeling and Estimation Techniques," Tech. Rep. RADC-TR-82-263, Oct. 1982.

[3] A.A. Abdel-Ghally, P.Y. Chan, and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," *IEEE Trans. on Software Engineering*, Vol. SE-12, No. 9, September 1986.

[4] J.J. Chilenski and S.P. Miller, "Applicability of Modified Condition/Decision Coverage to Software Testing," *Software Engineering Journal*, Sept. 1994.

[5] A.L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Models for Software Reliability and Other Performance Measures," *IEEE Trans. on Reliability*, Vol. R-28, No. 3, pp. 206-211, August 1979.

[6] A.L. Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," *IEEE Trans. on Software Engineering*, vol. SE-11, No. 12, December 1985.

[7] A.L. Goel and K. Okumoto, "A Non-Homogeneous Poisson Process Model for Software Reliability and Other Performance Measures," *Technical Report,* no 79-1, Dept. of IE and OR, Syracuse University, 1979.

[8] J.R. Horgan, S. London, and M.R. Lyu, "Achieving Software Quality with Testing Coverage Measure," *IEEE Computer*, pp. 60-69, Sept. 1994.

[9] M. Hutchins, H. Foster, T. Goradia and T. Ostrand, " Experiments on the Effectiveness of Dataflow-and Control-flow-based Test Adequacy Criteria," *Proc. of*

Table 1. Error Sum of Squares between Estimated and Observed Mean Value Functions

Coverage Function	Data Set 1	Data Set 2	Data Set 3	Data Set 4
Exponential	130.14	324.00	7336.2	2133.3
S-shaped	72.12*	3422.3	41365	33633
Weibull ($\gamma - 0.65$)	-	-	841.83*	-
Weibull ($\gamma - 0.70$)	-	-	939.96	-
Weibull ($\gamma - 0.75$)	-	-	1380.1	-
Weibull ($\gamma - 0.80$)	-	296.32	2180.1	-
Weibull ($\gamma - 0.85$)	-	234.17	3457.7	-
Weibull ($\gamma - 0.90$)	152.38	216.81*	5398.4	2351.4
Weibull ($\gamma - 0.95$)	140.72	245.15	7392.2	2015.6*
Weibull ($\gamma - 1.05$)	120.19	440.58	34701	2714.4
Weibull ($\gamma - 1.10$)	111.49	590.19	-	3798

* identifies best estimates of the mean value function.
- indicates an unreasonable fit to the observed interfailure times.

Table 2. *u*-plot Kolmogorov Distance

Coverage Function	Data Set 2	Data Set 3	Data Set 4
Exponential	0.1007*	0.3431	0.0986
S-shaped	0.3490	0.5862	0.2376
Weibull ($\gamma - 0.65$)	-	0.1330*	-
Weibull ($\gamma - 0.70$)	-	0.1728	-
Weibull ($\gamma - 0.75$)	-	0.1974	-
Weibull ($\gamma - 0.80$)	0.1634	0.2220	-
Weibull ($\gamma - 0.85$)	0.1674	0.2460	-
Weibull ($\gamma - 0.90$)	0.1248	0.2739	0.2028
Weibull ($\gamma - 0.95$)	0.1123	0.2991	0.1139
Weibull ($\gamma - 1.05$)	0.1217	0.2998	0.0866*
Weibull ($\gamma - 1.10$)	0.1472	-	0.0979

* identifies the coverage function with best predictive capability.
- indicates an unreasonable fit to the observed interfailure times.

Intl. Conference on Software Engineering, 1994.

[10] R. Jacoby and K. Masuzawa, "Test Coverage Dependant Software Reliability Estimation by the HGD Model," *3rd Intl. Symposium on Software Reliability Engineering*, 1992.

[11] P.A. Keiller, B. Littlewood, D.R. Miller and A. Sofer, "Comparison of Software Reliability Predictions," in *Dig. FTCS 13 (13th Int. Symp. Fault-Tolerant Comput.*, 1983, pp.128-134.

[12] J.D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[13] J.D. Musa, *Software Reliability Data*, report and database available from Data and Analysis Center for Software, Rome Air Development Center, Rome, NY.

[14] M. Ohba, "Software Reliability Analysis Models," *IBM J. Res Develop*, Vol. 28, No. 4, July 1984.

[15] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, Prentice Hall, 1982.

[16] E.J. Weyuker and P.G. Frankl, "An Applicable Family of Dataflow Testing Criteria," *IEEE Trans. Software Engineering*, Vol. SE-14, No. 10, Oct. 1988.

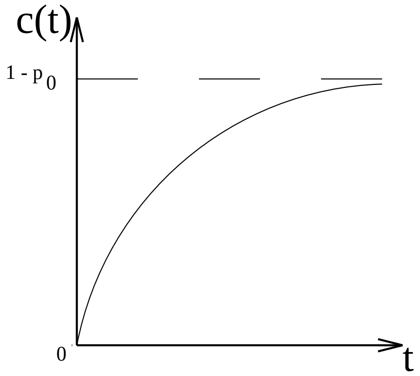

Figure 1. A defective coverage function: $c(t) = (1-p)c^c(t)$, where $c^c(t)$ is non-defective

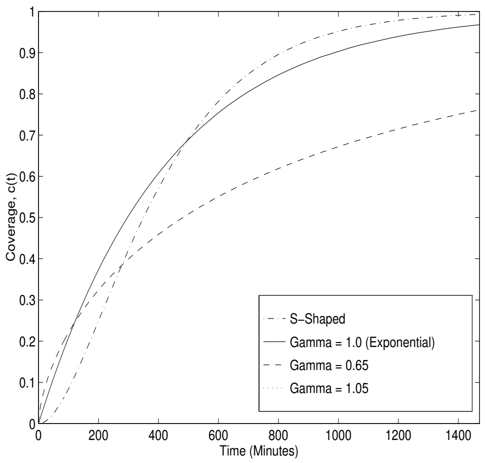

Figure 2. Typical Coverage Functions

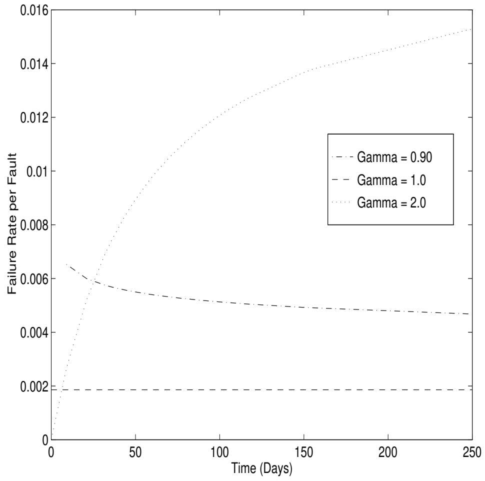

Figure 3. Typical Hazard Functions

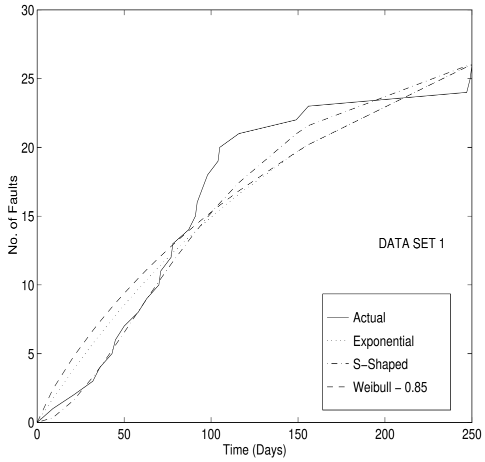

Figure 4. Mean Value Functions

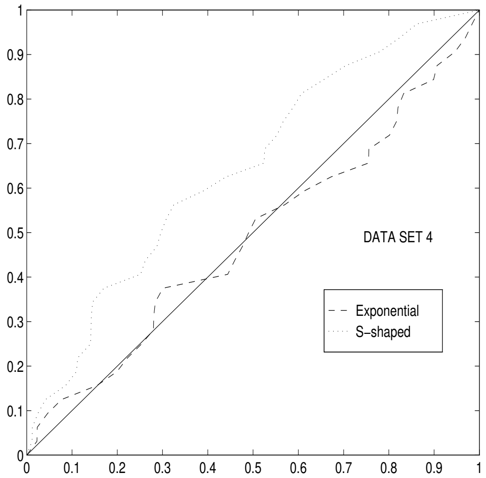

Figure 5. u-plot