Software Reliability Modeling with Test Coverage: Experimentation and Measurement with A Fault-Tolerant Software Project

Xia Cai and Michael R. Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong
{xcai, lyu}@cse.cuhk.edu.hk

Abstract

As the key factor in software quality, software reliability quantifies software failures. Traditional software reliability growth models use the execution time during testing for reliability estimation. Although testing time is an important factor in reliability, it is likely that the prediction accuracy of such models can be further improved by adding other parameters which affect the final software quality. Meanwhile, in software testing, test coverage has been regarded as an indicator for testing completeness and effectiveness in the literature.

In this paper, we propose a novel method to integrate time and test coverage measurements together to predict the reliability. The key idea is that failure detection is not only related to the time that the software experiences under testing, but also to what fraction of the code has been executed by the testing. This is the first time that execution time and test coverage are incorporated together into one single mathematical form to estimate the reliability achieved. We further extend this method to predict the reliability of fault-tolerant software systems. The experimental results with multi-version software show that our reliability model achieves a substantial estimation improvement compared with existing reliability models.

Keywords: *software reliability modeling, software testing, test coverage, fault-tolerant software.*

1 Introduction

As the key factor in software quality, software reliability quantifies software failures. Defined as the probability that a software system does not fail in a specified period of time in a specified environment, software reliability has become the most essential ingredient in customer satisfaction [9]. As a result, many analytical models have been proposed for software reliability estimation. The time-domain models, also called software reliability growth models (SRGM), have drawn most attention. These software reliability models use the failures collected in testing phases to predict the failure occurrences in the operational environment. There are two classes of basic data used in traditional SRGMs: 1) failures per time period; and 2) time between failures. A number of reliability models have been proposed to illustrate various distributions between failure/time and reliability, including some well-known models, e.g., the Goel-Okumoto (G-O) and Musa-Okumoto (M-O) models [9].

Although some of the historical SRGMs have been widely adopted to predict software reliability [9], it is likely that the prediction accuracy of these models can be further improved by adding other important factors affecting the final software quality [5, 6, 11, 13]. Test coverage is believed to be one of such factors in some of the previous studies [1, 7, 12]. In particular, to incorporate the effect of test coverage on reliability in traditional software reliability models, [5] proposes a technique using both time and test coverage measurement for reliability prediction. Basically, this approach reduces the execution time by a parameterized factor when the test case neither increases test cover-

1071-9458/07 $25.00 © 2007 IEEE
DOI 10.1109/ISSRE.2007.17

age nor causes a failure. Experiments show that the adjusted G-O and M-O models with such time reduction achieve more accurate predictions than the original ones. In line with other studies [8, 14], our own previous studies [2, 3, 10] have also found that test coverage has certain effects on software reliability.

Most of the existing software reliability models are based in the time domain, i.e., using either the elapsed time between software failures or the number of failures occurring over a specified execution time period [9]. However, examination of testing procedure suggests that execution time should not be the only factor that affects the failure behavior of the software. For example, if the testing sequence within a test set is changed, the time between failures or the number of failures within a certain time period may also change. In this situation, although the same test set is being executed on the same software system, different reliability predictions would be made by the traditional time-domain reliability growth models.

In this paper, we propose a novel method to integrate time and test coverage measurements together to predict the reliability. Before that, we first formulate the relationship between the number of failures and the test coverage achieved in test cases using two simplified models. The key idea of the new reliability model is that the reliability of a software system is not only affected by the testing time it undergoes, but also the completeness of the testing. Thus the reliability prediction is composed of two parts: the estimation from both execution time and test coverage. The two models we proposed, together with other existing coverage functions, can be used to describe the effect of coverage on reliability. For the effect of time on reliability, distributions from traditional SRGMs can be adopted for the estimation.

In literature, several models have been proposed to formulate the relationship between the number of failures/faults and test coverage achieved, using various distributions. [13] suggests that this relation follows a variant of the Rayleigh distribution, while [11] derives a result that the relationship can be expressed as a logarithmic-exponential formula, based on the assumption that both defect coverage and test coverage follow the Musa-Okumoto logarithmic growth model with respect to execution time. [6] assumes that coverage is a continuous monotonic non-decreasing function of testing time, and obtain a linear relationship between number of failures and test coverage achieved at time t.

In the following, we discuss the two coverage models in Section 2. The proposed reliability model is formulated in Section 3 and evaluated in Section 4. Finally, Section 5 elaborates the results and Section 6 concludes the paper.

2 Two Models of Defect Coverage and Test Coverage

2.1 A Hyper-exponential Model

According to our previous observations reported in [3], the relationship between the number of faults detected and test coverage achieved varies under different testing strategies. Based on this, we make the following assumptions:

1. For the relationship between defect coverage and test coverage, there are K classes on the whole test set, representing the different natures of the various testing strategies;

2. Within each class, the fault detection rate with respect to coverage is proportional to the number of faults remaining undetected;

3. A fault, when found, is corrected instantaneously without introducing new faults.

Following these assumptions, the fault detection rate with respect to coverage within each class is:

$$\frac{dF_c}{dc} = \beta \cdot F_r = \beta \cdot (N - F_c)$$

where F_c is the current cumulated number of faults detected when coverage c is achieved (c ranges from 0 to 1, and 1 representing complete coverage of the code), F_r is the number of residual faults, N is the total number of faults that are detectable by the current testing strategy, and β is a constant.

Solution of this differential equation in the range of $0 \leq c \leq 1$, under initial condition $F_0 = 0$, gives the following:

$$F_c = N(1 - e^{-\beta c}) \tag{1}$$

From the assumptions, each class follows the non-homogeneous Poisson process (NHPP) model with its own parameters [9]. So on the whole test set, the expected cumulated number of faults detected with coverage c is:

$$F_c = \sum_{i=1}^{K} N_i(1 - e^{-\beta_i c}) \quad (2)$$

where K is the number of classes. Notice that if $K = 1$ we have the NHPP model. Moreover, as N_i represents the expected total number of faults to be eventually detected in each class, the summation $\sum_{i=1}^{K} N_i$ is the total number of faults that will be detected under various testing strategies.

Since the failure intensity function λ_c is the derivative of F_c, we therefore have

$$\lambda(c) = \sum_{i=1}^{K} N_i \beta_i e^{-\beta_i c} \quad (3)$$

The two parameters in each class can be estimated using the maximum likelihood estimation (MLE) method or least-squares estimation (LSE), using the failure data and coverage information under that particular testing strategy.

2.2 A Beta Model

Unlike the Hyper-exponential model presented above, following the well-known G-O reliability growth model, we assume that both fault coverage and test coverage follow the NHPP model with respect to execution time, i.e.:

$$F_c(t) = N_1(1 - e^{-b_1 t}) \quad (4)$$

where $F_c(t)$ is the number of cumulated faults detected at time t, N_1 is the expected number of faults detected eventually, and b_1 is a constant.

Similarly, we have

$$c(t) = N_2(1 - e^{-b_2 t}) \quad (5)$$

where c(t) is the cumulated test coverage achieved at time t, N_2 is the ultimate test coverage that can be achieved by testing, and b_1 is a constant.

From (5), we can derive the formula for time t,

$$t = -\frac{1}{b_2} log(1 - \frac{c}{N_2})$$

Substituting t in (4), we get

$$F_c = N_1[1 - (1 - \frac{c}{N_2})^{\alpha}] \quad (6)$$

where $\alpha = b_1/b_2$.

From (6), the relationship between cumulated detected faults and test coverage follows a Beta distribution, where $\frac{c}{N_2} < 1$. Similarly, the parameters N_1, N_2 and α can be estimated by MLE or LSE methods using fault and coverage data collected during acceptance testing and operational testing.

2.3 Empirical Evaluation

In the year of 2002 we formed 34 independent programming teams at the Chinese University of Hong Kong to design, code, test, evaluate, and document a critical application taken from industry - the RSDIMU project [10]. Each team was composed of 4 senior-level undergraduate Computer Science students for a 12-week long project in a software engineering course. In this CUHK-RSDIMU project, 1200 test cases were designed and executed in the acceptance test. Failure data as well as the test coverage achieved in the testing for all these programs were collected. Detailed project and testing information can be found in [10, 3].

Here we apply the failure and coverage data in the CUHK-RSDIMU project to evaluate the two simplified models above. The LSE method is used to estimate the parameters in the models.

First of all, we evaluate the NHPP model as well as the hyper-exponential model. The estimated parameters and the sum of squared errors (SSE) are listed in Table 1. This shows that the NHPP model does not fit the failure/coverage data very well, although the SSE is slightly smaller for the larger value of N and the smaller value of β. If hyper-exponential modeling is applied on the six testing regions, following the underlying design strategies of various test cases illustrated in [3], it can be noted that the SSEs of Region II to VI are considerably smaller. Region I combines all the test cases which target basic functions in the programs under test. This is why Region I exhibits such a high diversity,

For the Beta model, if we assume the ultimate test coverage N_2 in (6) is 100%, we obtain

$$F_c = N_1[1 - (1 - c)^{\alpha}] \quad (7)$$

Table 1. Estimated parameters in the coverage models

Model	N	β/α	SSE
NHPP	5467	0.096	118200
Hyper-exponential	4087	–	23928
Region I	1989	0.256	22195
Region II	476	1.97	133
Region III	411	3.29	1315
Region IV	406	3.75	66
Region V	414	3.77	219
Region VI	391	21.3	1.01e-009
Beta	1101	0.303	38365

Hence, using the LSE method, we can derive the following parameters for Beta model: $N_1 = 1101$, and $\alpha = 0.303$, see Table 1. The SSE in this estimation is smaller than the NHPP model, but larger than the hyper-exponential model. A comparison of the NHPP model, the Hyper-exponential model and the Beta model is shown in Figure 1.

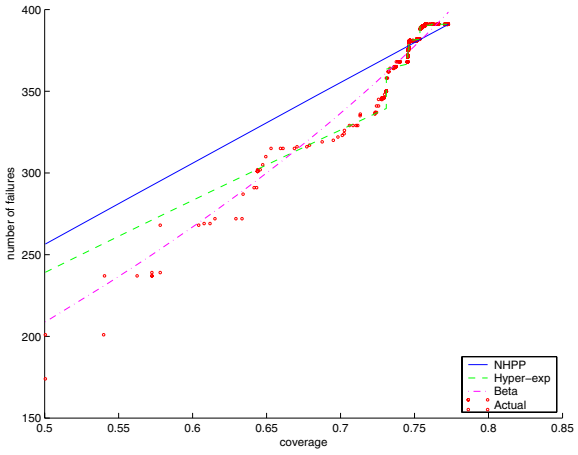

Figure 1. Comparison of the NHPP, Hyper-exponential and Beta estimation

3 A New Software Reliability Model

Here we propose a new reliability model which aims to predict the reliability performance using time between failures and coverage measurement together. The detailed assumptions and model form are illustrated as follows.

3.1 Assumptions

Our new reliability model is based on the following assumptions:

1. The number of failures revealed in testing is related to not only the execution time, but also the test coverage achieved by the current test set;
2. The failure rate with respect to time and test coverage together is a parameterized combination of those with respect to time or coverage alone;
3. The probabilities of failure with respect to time and coverage are not independent, but affect each other exponentially.

According to these assumptions, the data requirements for implementation of this model are: the time between failures, or the actual sequences of test cases in which the software failed, and the cumulated coverage measurement achieved by the whole test set.

3.2 Model Form

From the assumptions above, we can derive the joint failure intensity function with respect to both time and coverage as follows:

$$\lambda(t,c) = \alpha_1\gamma_1 e^{-\gamma_1 c}\lambda_1(t) + \alpha_2\gamma_2 e^{-\gamma_2 t}\lambda_2(c) \qquad (8)$$

where $\lambda(t,c)$ is the joint failure intensity function, $\lambda_1(t)$ is the failure intensity function with respect to time, and $\lambda_2(c)$ is the failure intensity function with respect to coverage. α_1, γ_1, α_2 and γ_2 are all parameters with the constraint of $\alpha_1 + \alpha_2 = 1$.

From the integral of the failure intensity function in (8), we can get the general expression of the expected cumulated number of failures when execution time is t, and cumulated coverage achieved is c:

$$\begin{aligned} F(t,c) = {} & \alpha_1(1 - e^{-\gamma_1 c})F_1(t) \\ & +\alpha_2(1 - e^{-\gamma_2 t})F_2(c) \end{aligned} \qquad (9)$$

where $F_1(t)$ is the expected cumulated number of failures relating to testing time t, and $F_2(c)$ is the expected

cumulated number of failures revealed by coverage c. The constraint $\alpha_1 + \alpha_2 = 1$ is maintained.

Since $\lambda_1(t)$ is the failure intensity function with respect to time, any existing distributions in well-known reliability models can be used, e.g., NHPP, the Weibull model, the S-shaped model and logarithmic Poisson models. Similarly, we can use any form such as the Hyper-exponential and Beta models proposed above for the failure intensity function with respect to coverage $\lambda_2(c)$.

To illustrate the detailed format of (8), if we use NHPP models for both time and coverage, we will get this joint failure intensity function:

$$\begin{aligned}\lambda(t,c) = \alpha_1\gamma_1 e^{-\gamma_1 c} N_1 \beta_1 e^{-\beta_1 t} \\ +\alpha_2\gamma_2 e^{-\gamma_2 t} N_2 \beta_2 e^{-\beta_2 c}\end{aligned} \tag{10}$$

From (9), we can obtain the expected cumulated number of failures when execution time is t, and cumulated coverage achieved is c:

$$\begin{aligned}F(t,c) = \alpha_1(1 - e^{-\gamma_1 c}) N_1 (1 - e^{-\beta_1 t}) \\ +\alpha_2(1 - e^{-\gamma_2 t}) N_2 (1 - e^{-\beta_2 c})\end{aligned} \tag{11}$$

On the other hand, if we use the Beta model for coverage, the joint failure intensity function will be :

$$\begin{aligned}\lambda(t,c) = \alpha_1\gamma_1 e^{-\gamma_1 c} N_1 \beta_1 e^{-\beta_1 t} \\ +\alpha_2\gamma_2 e^{-\gamma_2 t} N_2 \beta_2 (1-c)^{\beta_2 - 1}\end{aligned} \tag{12}$$

The expected cumulated number of failures is:

$$\begin{aligned}F(t,c) = \alpha_1(1 - e^{-\gamma_1 c}) N_1 (1 - e^{-\beta_1 t}) \\ +\alpha_2(1 - e^{-\gamma_2 t}) N_2 [1 - (1-c)^{\beta_2}]\end{aligned} \tag{13}$$

We can prove the joint density function $f(t,c)$ derived from (8) by calculating its theoretical integral with respect to t and c and getting a result of 1, as follows:

$$\begin{aligned}\int_0^\infty \int_0^\infty f(t,c)dtdc &= \int_0^\infty \int_0^\infty \alpha_1\gamma_1 e^{-\gamma_1 c} f_1(t) dtdc \\ &\quad + \int_0^\infty \int_0^\infty \alpha_2\gamma_2 e^{-\gamma_2 t} f_2(c) dtdc \\ &= \int_0^\infty \alpha_1\gamma_1 e^{-\gamma_1 c} dc + \int_0^\infty \alpha_2\gamma_2 e^{-\gamma_2 t} dt \\ &= \alpha_1 + \alpha_2 = 1\end{aligned}$$

4 Experimental Evaluation

In the CUHK-RSDIMU project, each team was required to use a Revision Control System (RCS) for source control, so that every code change of each program file could be recorded. Thus software faults found during each stage were identified. These faults were then injected into the final program versions to create mutants, each containing one programming fault. We selected 21 program versions for detailed investigation, and created 426 mutants according to certain generation strategies [10]. We disqualified the other 13 versions as their developers did not follow the development and coding standards which were necessary for generating meaningful mutants from their projects.

As we know, the CUHK-RSDIMU project is one of the largest fault-tolerant software projects with real-world application. The 426 mutants generated are real faults occurring in the development, rather than hypothetical faults injected into the programs. The other important characteristics of this project is the collection of test coverage during testing, including block coverage, decision coverage, C-Use and P-Use, which makes it possible for the investigation of the relationship between time, coverage and failure data.

4.1 Experimental Setup

To collect the time, coverage and failure data for reliability models with our own multi-version program versions and mutants, we use a super-program for testing and evaluation. The concept of a super-program was first proposed in [4] in order to make use of the testing data of fault-tolerant software for reliability estimation.

In our experiment, the super-program is composed of all the 21 program versions which contain 426 mutants, being treated as 426 faults or failures. The coverage is measured against the super-program.

The testing procedure is described as follows:

1. Initialize the testing pool, which contains the whole acceptance test set or operational test set;
2. Select a test case randomly from the testing pool;
3. Run the super-program according to one of three different testing strategies:

(a) Run all the mutants at the same time, find those failing and delete them;

(b) Select a program version randomly, run all the mutants within this version, record the mutants failed, and remove them from the super-program;

(c) Select one mutant from all the mutants in all versions, remove it if it fails, otherwise go to step 2;

4. Remove the current test case from the testing pool, and go to step 2.

From the three different selection strategies for program versions and mutants, we will get three different testing results. These testing data can be applied to this new reliability model for evaluation. Performance comparisons with other well-known reliability models, such as G-O, M-O, Musa Basic model, etc, can also be made based on the experimental data. Meanwhile, the estimation accuracy under three different selection strategies can be further investigated..

4.2 Estimation Method

In this experimental evaluation, we adopt the first testing strategy out of the three stated above, i.e., run all the mutants at the same time and remove those failing. For the other two testing strategies, we will evaluate and compare their performance with the first one in our later empirical study.

To estimate the parameters in our reliability model, we have to deal with different reliability models with respect to time and coverage separately. To make our evaluation clearer, we adopt the NHPP growth model for execution time, and the exponential or Beta model for test coverage at the different failure rates. Other failure rates can also be adopted for further evaluations. Least-squares estimation (LSE) method is used for parameter estimation in our experiment.

For each of the evaluations, we use two different methods as follows:

Method A: first, the parameters in $F_1(t)$ and $F_2(c)$ are estimated separately. After these parameters are determined, the other parameters (α_1, γ_1, α_2, and γ_2) are optimized using the joint failure rate;

Method B: identify the mathematical forms of $F_1(t)$ and $F_2(c)$, and then optimize all the parameters in (9) together according to existing experimental data.

In the two estimation methods, method A seems more reasonable, since it sets the failure rates first and estimates the parameters of dependencies on the basis of determined failure rates. However, method B is also a necessary complement, as it tries to capture the dependency between time and coverage together with their different failure rates according to the current empirical data. In our evaluation, in order to make the results more convergent, we always set the initial values in method B as the parameters estimated separately before.

In our evaluation study, we conduct comparisons between our model (using methods A and B) with some well-known time-based reliability models such as the NHPP, Weibull and S-Shaped models, whose expressions are listed in Table 2.

On the other hand, we also collect existing coverage-based reliability models, study their performance, and list their formula for comparison; see Table 3.

4.3 Comparison with existing SRGMs

In this evaluation, we compare the performance of other existing time-based SRGMs - namely, the NHPP, Weibull and S-Shaped models - with our model with different time and coverage distributions.

Supposing the defect coverage and test coverage follows the exponential relationship as shown in (1), we have the cumulative distribution function (cdf) expression as in (11). From LSE estimation for time and coverage separately, we get the parameters N_1, β_1 for time, and N_2, β_2 for coverage. Using these known parameters and failure data in the experiment, we get the other three parameters using least-squares estimation, as listed in Table 4 and illustrated in Figure 2. Note that in Figure 2, the unit of time axis is second with the assumption that each test case takes 0.1 second for execution. The NHPP SRGM with respect to execution time is shown in Figure 3.

Table 4 shows that our estimation method A affords an improvement compared with the NHPP model, while estimation method B improves the estimation performance significantly. This result supports our

Table 2. Comparison of reliability models

Model	Cumulated number of failures: F	Failure intensity function: λ
NHPP	$N(1-e^{-\beta t})$	$N\beta e^{-\beta t}$
Weibull	$N(1-e^{-\beta t^{\gamma}})$	$N\beta\gamma e^{-\beta t^{\gamma}} t^{\gamma-1}$
S-Shaped	$N[1-(1+\beta t)e^{-\beta t}]$	$N\beta^2 t e^{-\beta t}$
Our model	$\alpha_1(1-e^{-\gamma_1 c})F_1(t)+\alpha_2(1-e^{-\gamma_2 t})F_2(c)$	$\alpha_1\gamma_1 e^{-\gamma_1 c}\lambda_1(t)+\alpha_2\gamma_2 e^{-\gamma_2 t}\lambda_2(c)$

Table 3. Comparison of coverage functions

Author	Cumulated number of failures: F
Vouk (1992)	$N[1-e^{-\beta(c-c_{min})^2})]$
Gokhale&Trivedi(1999)	$ac(t)$ (where c(t) is the coverage function), e.g., $a\cdot\frac{(\lambda t)^k}{1+(\lambda t)^k}$
Chen et al. (2001)	Using coverage as a reduction parameter for testing time
Malaiya et al. (2002)	$\alpha_0\cdot\log[1+\alpha_1(e^{\alpha_2 c}-1)]$
Our model	$\alpha_1(1-e^{-\gamma_1 c})F_1(t)+\alpha_2(1-e^{-\gamma_2 t})F_2(c)$

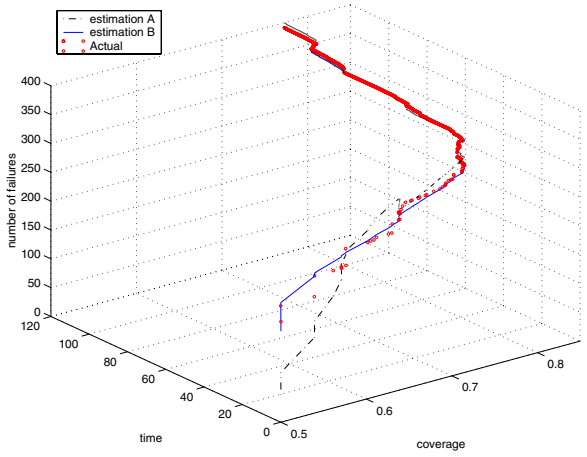

Figure 2. Reliability modeling with exponential failure rates

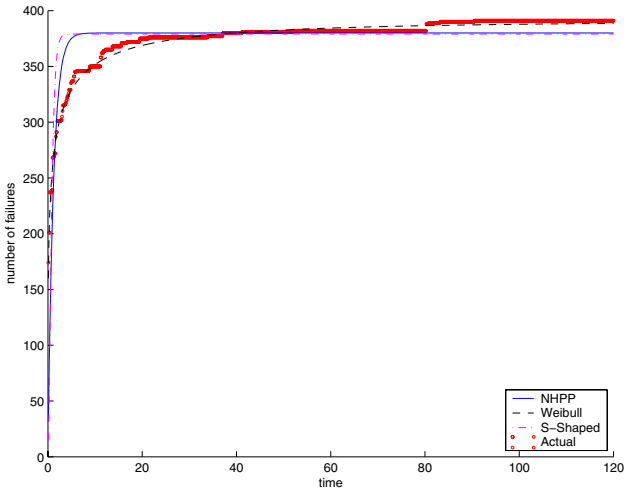

Figure 3. Reliability modeling with NHPP time relationship

previous observation: software failure behavior is related not only to testing time, but also to test coverage or completeness. Although in our case, α_2 is a negative number since α_1 is larger than 1, both time and coverage contribute to the number of failures detected in the testing, and to the ultimate final reliability.

Supposing the failure/coverage relationship follows the equation (7), we have the cdf expression shown in (13). Again, we use two estimation methods described in the previous section and compare their estimation performance in Table 5 and Figure 4.

From Figure 2 and Figure 4, it can be observed that the reliability estimation using our new reliability modeling is more accurate than that of NHPP model, especially for method B.

The comparison results between method A and B and the Weibull model itself can be found in Table 6[1].

[1]Please note that in Table 6, there are nine parameters with method A since α_2 can be derived from α_1; there are eight parameters with method B since α_1 and N_1 are combined together for method B. This is also applicable to the following Tables.

Table 4. Estimated reliability parameters for exponential coverage model

Method	α_1	γ_1	N_1	β_1	γ_2	N_2	β_2	SSE
A	-1.3844	3.0819	380	0.87	1.5110	1475	0.39	93849
B	1.7713	0.824	380	11.716	0.121	1475	-0.082	14130
NHPP Model	-	-	380	0.87	-	-	-	279230

Table 5. Estimated reliability parameters for the Beta coverage model

Method	α_1	γ_1	N_1	β_1	γ_2	N_2	β_2	SSE
A	0.0407	16.097	380	0.87	19.516	1101	0.303	36825
B	0.0565	20.182	380	0.098	21.138	1101	0.305	25712
NHPP Model	-	-	380	0.87	-	-	-	279230

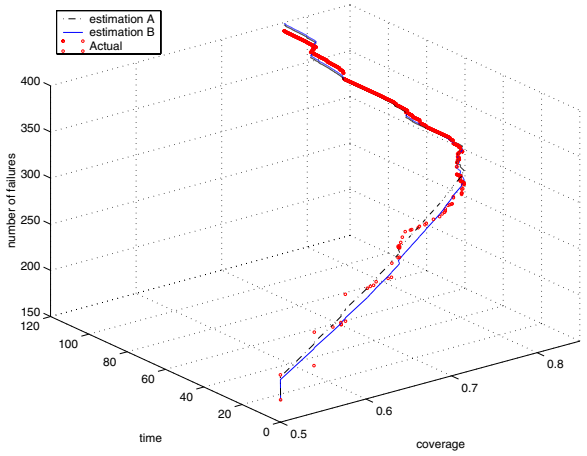

Figure 4. Reliability modeling with Beta coverage relationship

A similar estimation improvement can be observed for our combined model for both method A and method B. The curve of Weibull model is shown in Figure 3.

The estimation comparisons with S-Shaped distributions for both time and coverage measurement are shown in Table 7. The S-Shaped model with time alone can be found in Figure 3. Surprisingly, we can see that, with our model, the estimation fits the actual data perfectly, which is a significant improvement compared with the original time-based S-Shaped reliability model.

As shown in the evaluation and comparison above, the advantage of our reliability model is that: this is the first time that test coverage information is combined into traditional time-based software reliability modeling as one input parameter. Our experimental results have shown that our reliability estimation is more accurate and flexibly able to be incorporated with any existing time or coverage-based model.

4.4 Comparison with other coverage-based models

We compare the fitness of various coverage-based models listed in Table 3 together with our Hyper-exponential and Beta coverage model. How these models fit the failure and coverage data we collected is illustrated in Figure 5.

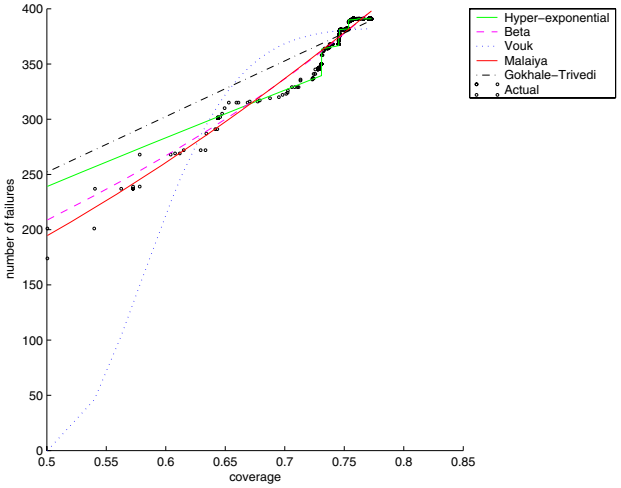

Figure 5. Comparisons with existing coverage functions

Table 6. Estimated reliability parameters for the Weibull coverage model

Method	Parameters	SSE
A	{0.74, 48.1, 391, 1.1, 0.32, 197.76, 465.8, 4.44, 3.37}	10631
B	{380,1.5, 1.0, 0.3, 1475, 27.3, 0.1, 0.5}	8516
Weibull	{391, 1.1, 0.32}	13101

Table 7. Estimated reliability parameters for the S-Shaped coverage model

Method	Parameters	SSE
A	{1.005, 0.0057, 379, 2.46, 1.447, 3515, 0.7}	0.9302
B	{380, 0, 1.0, -0.05, 1.6, -0.06}	$4 \cdot 10^{-9}$
S-Shaped	{379, 2.46}	392180

We further estimate the parameters in these models using the least-squares method. The parameter estimation and SSEs are listed in Table 8. Here we also give the parameters of our model with Hyper-exponential, Beta and Weibull functions respectively, in which all the parameters are estimated together (method B). All the parameters in Table 8 follow the same order as those in Table 3.

From Table 8, we note that our model, which combines time and coverage functions, performs better than other coverage-based models for all three distributions: Hyper-exponential, Beta and Weibull. In particular, the model with the Weibull distribution exhibits the best accuracy. Malaiya et al. [11] report a model with a similar SSE value, while Vouk's and Gokhale-Trivedi's models have an SSE with a larger magnitude. One of the underlying reasons may be that our empirical data does not exhibit a linear relationship between coverage and time, as assumed in Gokhale-Trivedi model. Although in their model, the linear parameter does not have to be constant, we use a constant linear relationship for simplicity here. Further empirical investigations of the performance comparison are needed, as long as the coverage and failure data can be collected for large-scale software systems.

5 Discussions

Based on the experimental evaluation above, we can see that the advantage of our proposed time- and coverage-based reliability model, which aims to predict the number of failures based on both time and coverage information. This is the first time that these two measurements for failure prediction have been merged in one single reliability model.

Furthermore, in our reliability model, although the effects of time and coverage on number of failures have been combined together, they are independent from each other. The two effects can be estimated or modeled separately, i.e., $F(t)$ and $F(c)$ can be either of the existing models or distributions that have been proposed for failure predictions based on time or coverage information alone. After these two functions have been estimated independently, the other parameters in our combined model can be estimated together. In other words, our model is highly flexible and compatible with other reliability models. On the other hand, our model is more demanding in terms of the empirical data, i.e., not only should the testing time and number of failures be available, but also the coverage measurements need to be collected during the testing phases. Unfortunately, this information is rarely available, hindering furthering empirical studies.

6 Conclusion

In this paper, we propose a new reliability model which integrates time and coverage measurements for reliability prediction. The key idea is that failure detection is not only related to the time that the software experiences under testing, but also how much the code fraction has been executed by the testing. This is the first time that execution time and test coverage are incorporated together into one single mathematical form to estimate the reliability achieved.

Our experimental results show that our reliability

Table 8. Performance of different coverage functions

Coverage function	Parameters	SSE
Vouk	[382, 82]	345410
Gokhale & Trivedi	[380,1.4,1.0]	594390
Malaiya et al.	[382, 0.16, 3.23]	35870
Our model(hyper-exp)	[1.7713,0.824,380,11.716,0.121,1475,-0.082]	14130
Our model(Beta)	[0.0565,20.182,380,0.098,21.138,1101,0.305]	25712
Our model(Weibull)	[380, 1.5, 1.0, 0.3, 1475, 27.3, 0.1, 0.5]	8516

model gives an more accurate estimation than some existing time- and coverage-based models with a significant improvement. However, the choice of coverage distribution does affect the final estimation performance with our experimental data.

In our future work, we will conduct further empirical evaluations for our model and other well-known reliability models. The other two testing strategies need to be employed for further comparison. More experimental data are also needed for this purpose.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4150/07E).

References

[1] P. G. Bishop. Estimating residual faults from code coverage. In *Proceedings of the 21st International Conference on Computer Safety, Reliability and Security*, pages 10–13, Catania, Italy, September 2002.

[2] X. Cai. *Coverage-Based Testing Strategies and Reliability Modeling for Fault-Tolerant Software Systems*. PhD thesis, The Chinese University of Hong Kong, Hong Kong, September 2006.

[3] X. Cai and M. R. Lyu. The effect of code coverage on fault detection under different testing profiles. In *ICSE 2005 Workshop on Advances in Model-Based Software Testing (A-MOST)*, St. Louis, Missouri, May 2005.

[4] M. H. Chen, M. R. Lyu, and E. Wong. Incorporating code coverage in the reliability estimation for fault-tolerant software. In *Proceedings 16th IEEE Symposium on Reliable Distributed Systems*, pages 45–52, Durham, North Carolina, October 1997.

[5] M. H. Chen, M. R. Lyu, and E. Wong. Effect of code coverage on software reliability measurement. *IEEE Transactions on Reliability*, 50(2):165–170, June 2001.

[6] S. Gokhale and K. S. Trivedi. A time/structure based software reliability model. *Annals of Software Engineering*, 8:85–121, 1999.

[7] M. Grottke. A vector markov model for structural coverage growth and the number of failure occurrences. In *Proceedings of the 13th International Symposium on Software Reliability Engineering*, pages 304–315, Annapolis, Maryland, November 2002.

[8] J. Horgan, S. London, and M. Lyu. Achieving software quality with testing coverage measures. *IEEE Computer*, 27(9):60–69, September 1994.

[9] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1996.

[10] M. R. Lyu, Z. Huang, K. S. Sze, and X. Cai. An empirical study on testing and fault tolerance for software reliability engineering. In *Proceedings 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'2003)*, pages 119–130, Denver, Colorado, November 2003.

[11] Y. K. Malaiya, N. Li, J. M. Bieman, and R. Karcich. Software reliability growth with test coverage. *IEEE Transactions on Reliability*, 51(4):420–426, December 2002.

[12] A. T. Rivers and M. A. Vouk. An empirical evaluation of testing efficiency during non-operational testing. In *Proceedings of the 4th Software Engineering Research Forum*, pages 111–120, 1995.

[13] M. A. Vouk. Using reliability models during testing with nonoperational profiles. In *Proceedings 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation*, pages 103–111, October 1992.

[14] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur. Effect of test set size and block coverage on the fault detection effectiveness. In *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pages 230–238, Monterey, CA, November 1994.