The Relationship Between Test Coverage and Reliability

Yashwant K. Malaiya[*]
Naixin Li
Jim Bieman[†]
Computer Science Dept.
Colorado State University
Fort Collins, CO 80523
malaiya@cs.colostate.edu

Rick Karcich
Bob Skibbe
StorageTek
2270 South 88th Street
Louisville, CO 80028-2286
(303) 673-6223
Rick_Karcich@stortek.com

Abstract

We model the relation among testing effort, coverage and reliability, and present a logarithmic model that relates testing effort to test coverage (block, branch, c-use or p-use). The model is based on the hypothesis that the enumerables (like branches or blocks) for any coverage measure have different detectability, just like the individual defects. This model allows us to relate a test coverage measure directly with defect coverage. Data sets for programs with real defects are used to validate the model. The results are consistent with the known inclusion relationships among block, branch and p-use coverage measures. We show how defect density controls time to next failure.

The model can eliminate the variables like test application strategy from consideration. It is suitable for high reliability applications where automatic (or manual) test generation is used to cover enumerables which have not yet been tested.

1 Introduction

Developers can increase the reliability of software systems by measuring reliability as early as possible during development. Early indications of reliability problems allow developers to correct errors and make process adjustments.

Reliability can be estimated as soon as running code exists. To quantify reliability during testing, the code (or portion of code) is executed using inputs randomly selected following an operational distribution. Then, a reliability growth model can be used to predict the amount of effort required to satisfy product reliability requirements.

The needs of early reliability measurement and modeling are not met by common testing practices. The focus of testing is on finding defects, and defects can be often found much faster by non-random methods [2]. Testing is directed towards inputs and program components where errors are more likely. For example, testing may be conducted to insure that particular portions of the program and/or boundary cases are covered.

Models that can measure and predict reliability based on the status of non-random testing are clearly needed. Reliability achieved will be affected by:

- the testing strategy: Test coverage may be based on the functional specification (black-box), or it may be based on internal program structure (white-box). Strategies can vary in their ability to find defects.
- the relationship between calendar time and execution time: The testing process can be accelerated through the possibly parallel, intensive execution of tests at a faster rate that would occur during operational use.
- the testing of rarely executed modules: Such modules include exception handling or error recovery routines. These modules rarely run, and are notoriously difficult to test. Yet, they are critical components of a system that must be highly reliable. Only by forcing the coverage of such critical components, can reliability be predicted at very high levels.

Intuition and empirical evidence suggests that test coverage must be related to reliability. Yet, the connection between structure based measurements, like test coverage, and reliability is still not well understood.

Ramsey and Basili [24] investigated different permutations of the same test set and collected data relat-

[*] Y. Malaiya and N. Li are partly supported by a BMDO funded project monitored by ONR

[†] J. Bieman is supported, in part, by the NASA Langley Research Center, the Colorado Advanced Software Institute (CASI), Storage Technology Inc, and Micro-Motion Inc. CASI is supported in part by the Colorado Advanced Technology Institute (CATI). CATI promotes advanced technology teaching and research at universities in Colorado for the purpose of economic development.

1071-9458/94 $4.00 © 1994 IEEE

ing the number of tests to statement coverage growth. They tried a variety of models to fit the data. The best fit was obtained using the Goel and Okumoto's exponential model (GO model).

Dalal, Horgan and Ketterring [6] examined the correlation between test coverage and the error removal rate. They give a scatter plot of the number of faults detected during system testing versus the block coverages achieved during unit testing. The plot shows that modules covered more thoroughly during unit testing are much less likely to contain errors.

Vouk [28] found that the relation between structural coverage and fault coverage is a variant of the Rayleigh distribution. He assumed that the fault detection rate during testing is proportional to the number of faults present in the software and test coverage values including block, branch, data-flow, and functional group coverage. Vouk's experimental results suggest the use of a more general Weibull distribution. It was observed that, in terms of error removal capability, the relative power of the coverage measures block:p-use:DUD-chains is 1:2:6.

Chen et al [5] add structural coverage to traditional time-based software reliability models (SRMs). Their model excludes test cases that do not increase coverage. The adjusted test effort data is used to fit existing time-based models to avoid overestimation from traditional time-based SRMs due to the saturation effect of testing strategies.

Assuming random testing, Piwowarski, Ohba and Caruso [22] analyze block coverage growth during function test, and derive an exponential model relating the number of tests to block coverage. Their model is equivalent to the GO model [24]. They also derive an exponential model relating the covering frequency to the error removal ratio. The utility of the model relies on prior knowledge of the error distribution over different functional groups in a product.

Frankl and Weiss [9] compare the fault exposing capability of branch coverage and data flow coverage criteria. They find that for 4 out of 7 programs, the effectiveness of a test in exposing an error is positively correlated with the two coverage measures. They observed complex relationships between test coverage growth and the probability of exposing an error for a test set. Since the 7 programs they used are very small and they only considered subtle errors, the result may not be directly applicable to practical software.

The Leone test coverage model given in [21] is a weighted average of four different coverage metrics achieved during test phases: lines of executable code, independent test paths, functions/requirements, and hazard test cases. The weighted average is used as an indicator of software reliability. The model assumes that full coverage of all four metrics implies that the software tested is 100% reliable. In reality, such software may have some remaining faults. A similar approach, but with different coverage metrics, was taken to provide a test quality report [23].

In this paper, we explore the connection between test coverage and reliability. We develop a model that relates test coverage to defect coverage. With this model we can estimate the defect density. With knowledge of the operational profile, we can predict reliability from test coverage measures.

2 Coverage of Enumerables

The concept of test coverage is applicable for both hardware and software. In hardware, coverage is measured in terms of the number of possible faults covered. For example, each node in a digital system can possibly be stuck-at 0 or stuck-at 1. A stuck-at test coverage of 80% means that the tests applied would have detected any one of the 80% faults covered.

In contrast, the number of possible software faults in not known. Test coverage in software is measured in terms of structural or data-flow units that have been exercised. These units can be statements (or blocks), branches, etc. as defined below:

- Statement (or block) coverage: the fraction of the total number of statements (blocks) that have been executed by the test data.
- Branch (or decision) coverage: the fraction of the total number of branches that have been executed by the test data.
- C-use coverage: the fraction of the total number of computation use (c-uses) that have been covered by one c-use path during testing. A c-use path is a path through a program from each point where the value of a variable is modified to each c-use (without the variable being modified along the path).
- P-use coverage: the fraction of the total number of p-uses that have been covered by one p-use path during testing. A p-use path is a path from each point where the value of a variable is modified to each p-use, a use in a predicate or decision (without modifications to the variable along the path).

When such a unit is exercised, it is possible that one or more associated faults may be detected. Counting the number of units covered gives us a measure of the extent of sampling. The defect coverage in software can be defined in an analogous manner; it is the fraction of actual defects initially present that would be detected by a given test set.

In general, test coverage increases when more tests are applied, provided that the test cases are not repeated and complete test coverage has not already been achieved. A small number of enumerables may not be reachable in practice. We assume that the fraction of such enumerables is negligible.

Figure 1 [31] shows the relationship among some well-known criteria as proven by Weyuker. If there is a directed path from criteria A to criteria B, then test sets that meet criteria A (achievement of complete coverage) are guaranteed to satisfy criteria B. Table 1 shows the maximum number of tests [31] to satisfy these criteria and the observed complexities [30] for some specific cases. The upper bound for all-du-paths was reached in one subroutine out of 143 considered by Bieman and Schultz [bisc89,bisc92].

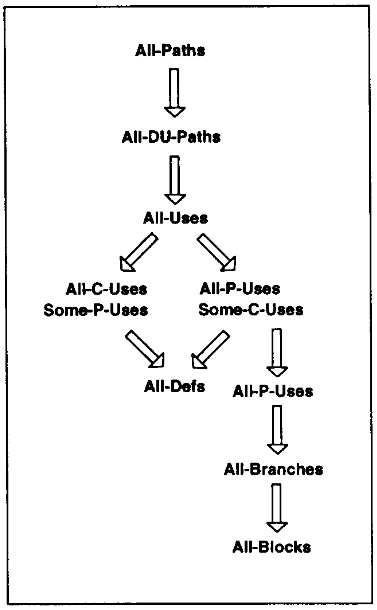

Figure 1: The subsumption relationships of different complete coverage criteria [31]

To keep the following discussion general, we will use the term *enumerable*. For branch coverage, the enumerables are branches, for defect coverage the enumerables are defects and so on. We use the term "enumerable-type" to imply defects, blocks, branches, c-uses or p-uses. We use superscript i, $i = 0$ to 4, to indicate one of the five types in this sequence: 0: defects, 1: blocks, 2:branches, 3: c-uses, 4: p-uses.

3 Detectability Profiles of Enumerables

The coverage achieved by a set of tests depends not only on the number of tests applied (or, equivalently, the testing time) but also on the distribution of *testability* values of the enumerables. A statement which is reached more easily is more testable. Such statements are likely to get covered (i.e. exercised at lease once) with only a small number of tests. Testability also depends on the likelihood that a fault that is reached actually causes a failure [27]. A statement that is executed only in rare situations has low testability. It may not get exercised by most of the tests that are normally applied. As testing progresses, the distribution of testability values will shift. The easy-to-test enumerables are covered early during testing, and are removed from consideration. The remaining enumerables include a larger fraction of hard-to-test enumerables. Thus, the growth of coverage will be slower.

Table 1: The complexity (test length) for achieving different coverage criteria [31]

Coverage Criterion	Upper bound	Observed
All-Blocks	$d+1$	
All-Branches	$d+1$	
All-P-Uses	$\frac{1}{4}(d^2+4d+3)$	$0.38d+3.17$
All-Defs	$m+(i \times n)$	
All-P-Uses/Some-C-Uses	$\frac{1}{4}(d^2+4d+3)$	
All-C-Uses/Some-P-Uses	$\frac{1}{4}(t^2+4d+3)$	$0.36d+2.82$
All-Uses	$\frac{1}{4}(d^2+4d+3)$	$0.39d+3.76$
All-DU-Paths	2^d	$0.49d+4.03$ 2^d
All-Paths	∞	

n: variables, m: assignments, i: input statements, d: binary decisions.

Definition: Detectability of an enumerable d^j_l is the probability that the l-th enumerable of type j will be exercised by a randomly chosen test.

The *detectability profile* is the distribution of detectability values in the system under test. The detectability profiles were introduced by Malaiya and Yang [15]. They have been used to characterize testing of hardware [29] and software [19]. A continuous version of the detectability profile was defined by Seth, Agrawal and Farhat [25]. For convenience, we use the normalized detectability profile (NDP) as defined below.

Definition: The discrete NDP for the system under test is given by the vector,

$$P^j = \{p^j_{d1}, p^j_{d2}, \ldots, p^j_{di}, \ldots, p^j_{du}\}$$

$$d^j_{i-1} < d^j_i < d^j_{i+1} \qquad (1)$$

where p^j_{di} is the fraction of all enumerables of type j which have detectability equal to di. Thus $p^2_{0.3}$ represents the fraction of all branches with detectability of 0.3. In Equation 1, du is the detectability value of unity (1), the highest value possible. $\sum_{di=0}^{1} p^j_{di} = 1$ since all fractions added will be unity.

A detectability value of 0 is possible, since a branch might be infeasible, or a defect might not be testable due to redundancy in implementation. Researchers have compiled detectability profiles of several digital circuits [15, 29] and software systems [26, 7].

If the number of enumerables is large, a continuous function can approximate the discrete NDP.

Definition: The continuous NDP, for the system under test is the function $p^j(x)$, $0 \leq x \leq 1$

$$p^j(x)dx = \frac{nr\_enumerables^j(x, x+dx)}{all\_enumerables^j} \qquad (2)$$

where $nr\_enumerables^j(x, x+dx)$ denotes all enumerables of type j with detectability values between x and $x + dx$.

$\int_0^1 p^j(x)dx = 1$, just like the discrete NDP case.

4 A one-parameter Model

The detectability profile gives the probability of exercising each enumerable. Hence, it can be used to calculate expected coverage when a given number of tests have been applied. Here, we assume that testing is random, i.e. any single test is selected randomly with replacement. Malaiya and Yang [15], and Wagner et al [29] show that the expected coverage of the enumerables of type j is

$$C^j(n) = 1 - \sum_{i=1}^{n}(1 - d_i^j)^n p_i^j \qquad (3)$$

provided testing is random. The same result holds for continuous NDP [25]

$$C^j(n) = 1 - \int_0^1 (1-x)^n p(x)dx \qquad (4)$$

Actual testing of software is more likely to be pseudo-random, since a test once applied will not be repeated. In such cases, random testing is an approximation. This approximation is reasonable, except when coverage approaches 100%.

Equations 3 and 4 give expected coverage. In a specific case, the coverage can be different. The central limit theorem suggests that results obtained should be close to these given by Equations 3 and 4 when a large number of vectors are applied.

The use of Equations 3 and 4 requires knowledge of detectability profiles. Obtaining exact detectability profiles requires a lot of computation. Discrete detectability profiles have been calculated for several small and large combinational circuits. Continuous detectability profiles for some benchmark circuits have been estimated [25]. However software systems are generally much more complex.

Fortunately, it is possible to obtain reasonable approximation for the detectability profiles. When one test is applied, the probability that an enumerable with detectability d_i^j will not be covered is $(1 - d_i^j)$. The probability that an enumerable will not be covered by n tests and thus remain a part of profile is $(1-d_i^j)^n$. Thus if the initial profile was given by Equation 1, the profile after having applied n tests, will be given by

$$P_n^j = \{p_{d1}^j(1 - d1^j)^n, p_{d2}^j(1 - d2^j)^n, ...\}$$

Equivalently the continuous profile is given by

$$p_n^j(x) = p_n(x)(1-x)^n$$

Thus enumerables with high testability are likely to get covered earlier. The profile will "erode" as testing progresses (see Figure 2). Enumerables with low testability get removed at a much lower rate, and thus will soon dominate. During much of the testing, the shape of the profile will appear like the bottom curve in Figure 2, regardless of the initial profile.

Available results for hardware components suggest that initial detectability profiles may be of the form

$$p^j(x) = (mj + 1)(1 - x)^{mj} \qquad (5)$$

where mj is a parameter. The factors $(mj + 1)$ ensure that the area under the initial profile curve is unity. By substituting the right hand side of Equation 5 in Equation 4, we get

$$C^j(n) = 1 - (mj + 1)\int_0^1 (1-x)^{mj+n}dx$$

$$= 1 - \frac{mj+1}{mj+n+1} = \frac{n}{mj+n+1} \qquad (6)$$

The curve given by Equation 6 matches the shape of experimental data. However it does not provide a good fit. One problem is that Equation 6 includes only a single parameter which can be adjusted for fitting. We can assume a more general initial detectability in Equation 5, involving two parameters, but even that may not be accurate, as we discuss in the next section. The approach considered next, yields a much better model.

5 A New Logarithmic Coverage Model

Random testing implies that a new test is selected regardless of the tests that have been applied thus far, and that tests are selected based only on the operational distribution. In actual practice, a test case is selected in order to exercise a functionality or enumerable that has remained untested so far. This process makes actual testing more directed and hence more efficient than random testing.

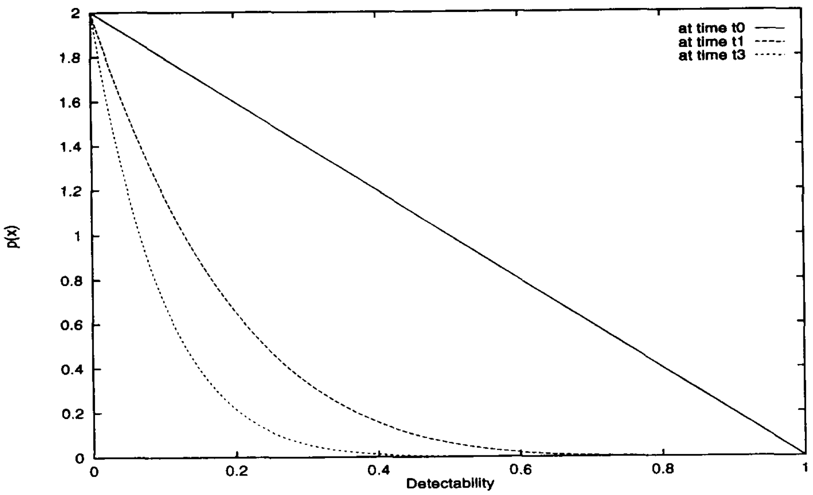

Figure 2: Distribution of detectability

Malaiya, von Mayrhauser and Srimani [18] show that this non-random process leads to a defect finding behavior described by the logarithmic growth model [17]. Their analysis gives an interpretation for the model parameters. The coverage growth of an enumerable-type depends on the detectability profile of the type and the test selection strategy. If the defect coverage growth in practice is described by the logarithmic model, it is likely that the coverage growth for other enumerable-types is also logarithmic. We suggest the following model.

$$C^i(t) = \frac{1}{N^i}\beta_0^i \ln(1+\beta_1^i t), \qquad C^i(t) \leq 1 \qquad (7)$$

where N^i is the total number of enumerables of type i, β_0^i and β_1^i are model parameters. If a single application of a test takes T_s seconds, then the time t, needed to apply n tests is nT_s. Substituting in 7,

$$C^i(n) = \frac{\beta_0^i}{N^i}\ln(1+\beta_1^i T_s n)$$

Defining b_0^i as $(\frac{\beta_0^i}{N^i})$ and b_1^i as $(\beta_1^i T_s)$, we can rewrite the above as,

$$C^i(n) = b_0^i \ln(1+b_1^i n), \qquad C^i(n) \leq 1 \qquad (8)$$

When $C^i = 1$, there are no more additional enumerables of that type to be found. With non-random testing a finite, although possibly large, number of tests are required to achieve 100% coverage of the feasible enumerables.

For defects ($i = 0$), the parameters β_0^0 and β_1^1 have the following interpretation [19].

$$\beta_0^0 = \frac{K^0(0)N^0(0)}{a^0 T_L} \qquad (9)$$

and

$$\beta_1^0 = a^0 \qquad (10)$$

where $K^0(0)$ is the exposure ratio at time t=0, T_L is the linear execution time and a^0 is a parameter that describes the variation in the exposure ratio.

Equation 8 relates coverage C^i to the number of tests applied. We use it to obtain an expression giving defect coverage C^0 in terms of one of the coverage metrics C^i, $i = 1$ to 4. Using Equation 8, we solve for n,

$$n = \frac{1}{b_1^i}[exp(\frac{C^i}{b_0^i}) - 1], \qquad i = 1 \ to \ 4$$

Substituting for C^0, again using Equation 8,

$$C^0 = b_0^0 \ln[1 + \frac{b_1^0}{b_1^i}(exp(\frac{C^i}{b_0^i}) - 1)], \qquad i = 1 \ to \ 4$$

Defining $a_0^i = b_0^0$, $a_1^i = \frac{b_1^0}{b_1^i}$ and $a_2^i = \frac{1}{b_0^i}$, we can write the above using three parameters as,

$$C^0 = a_0^i \ln[1 + a_1^i(exp(a_2^i C^i) - 1)] \qquad i = 1 \ to \ 4 \qquad (11)$$

Equation 11 gives a convenient three-parameter model for defect coverage in terms of a measurable test coverage metric. Equation 11 is applicable for only $C^0 \leq 1$. It is possible to approximate Equation 11 using a linear relation, but it would be accurate for only a small range.

6 Analysis of Data

We evaluate the proposed model, as given by Equations 8 and 11, using four data sets. The first data set, DS1, is from a multiple-version automatic airplane landing system [14]. It was collected using the ATAC tool developed at Bellcore. The twelve versions have a total of 30,694 lines. The data used is for integration and acceptance test phases, where 66 defects were found. One additional defect was found during operational testing. The second data set, DS2, is from a NASA supported project implementing sensor management in inertial navigation system [28]. For this program, 1196 test cases were applied and 9 defects were detected. The third data set, DS3, is for a simple program used to illustrate test coverage measures [1]. The fourth data set, DS4, is from an evolving software system containing a large number of modules.

Table 2: Summary table for DS1
(total 21,000 tests applied)

	Blocks	Decisions	c-uses	p-uses	Defects
	i=1	i=2	i=3	i=4	i=0
Total enum.	6977	3524	8851	4910	67
Final cov.	91.8%	83.9%	91.7%	73.5%	98.4%
b_0^i	0.031	0.049	0.036	0.041	0.184
b_1^i	2E+8	1234	3.4E+6	2439	0.01
LSE	5.7E-4	3.5E-5	5.8E-4	8.1E-5	7.3E-7

Table 3: Summary table for DS2
(total 1196 tests applied)

	Blocks	Branches	c-uses	p-uses	Defects
	i=1	i=2	i=3	i=4	i=0
Final cov.	89%	84%	76%	61%	90%
a_0^i	1.31	0.46	0.23	0.29	
a_1^i	1.8E-3	4.6E-3	9.11E-7	5.2E-3	
a_2^i	6.95	3.84	23.12	13.46	
LSE	0.017	0.018	0.041	0.025	

The first data set and the results from it are summarized in Table 2. The first row gives the total number of enumerables for all versions. The second row gives the average coverage when 21,000 tests had been applied. The values of the estimated parameters b_0^i and b_1^i and the least square error are given in the rows below. The model given by Equation 8 fits the data well. The data shows that $C^1 > C^2 > C^4$. This relationship is expected. Complete decision coverage implies complete block coverage, and complete p-uses coverage implies complete decision coverage [2, 8, 20]. The c-uses coverage has no such relation relative to the other metrics. Indeed the data shows that while $C^3 < C^1$ at the beginning of testing, near the end of testing C^3 is almost equal to C^1.

Table 3 summarizes the result for DS2. Nine faults were revealed by application of 1196 tests; we assume that one fault (i.e. 10%) is still undetected. In spite of the small number of faults, the model given in Equations 11 fits the data well.

Figure 3 shows the correlation of other test coverage measures C^2, C^3 and C^4 with block coverage C^1. As we expect, branch coverage, and to a lesser extent p-use coverage, are both strongly correlated with block coverage. The correlation with c-use coverage is weaker. Figure 4 shows actual and computed values for fault coverage. The computed values have been obtained using branch coverage and Equation 11. At 50% branch coverage the fault coverage is still quite low (about 10%), however with only 84% branch coverage, 90% fault coverage is obtained. The branch coverage shows saturation at about 84%. This supports the view that 80% branch coverage is often adequate [10].

Figure 5 is a scatter plot of computed values of defect coverage against actual values. The computed values are from the number of tests and Equation 8 (traditional reliability growth modeling), and test coverage measures C^1, C^2, C^3 and C^4 using Equation 11. The calculated values are all quite close, showing that coverage based modeling can replace time-based modeling.

Table 4: Summary table for DS3
(total 16 tests applied)

	Blocks	Branches	c-uses	p-uses
	i=1	i=2	i=3	i=4
Total enums	12	10	10	26
Final cov.	100%	100%	100%	93%
b_0^i	0.06	0.11	0.11	0.12
b_1^i	8.4E5	769	561	162
LSE	2E-5	1.7E-3	1.6E-3	2.3E-3

Table 4 shows similar results for a very small illustrative program. No defects were involved. However, this again demonstrates the applicability of our modeling scheme. We see that $C^1 \geq C^2 \geq C^4$. The c-use coverage again behaves differently.

In evolving programs, significant changes are being made while testing is in progress. Because new modules are being added, new defects as well as noncovered enumerables are also being added. The coverage obtained by a test set can actually go down in some cased. From DS4, (see Figure 6), the linear correlation between coverage measures can still be applicable. The data used here covers an intermediate phase of the process. The analysis of evolving programs is more complex and is the subject of future research.

7 Model Parameters

Researchers find that the logarithmic model works best among other two-parameter models [16], however interpretation of its parameters is difficult. One interpretation given by Malaiya et al [18] is described by Equations 9 and 10. The same interpretation may be applicable for enumerables other than defects. The first parameter of Equation 8 is,

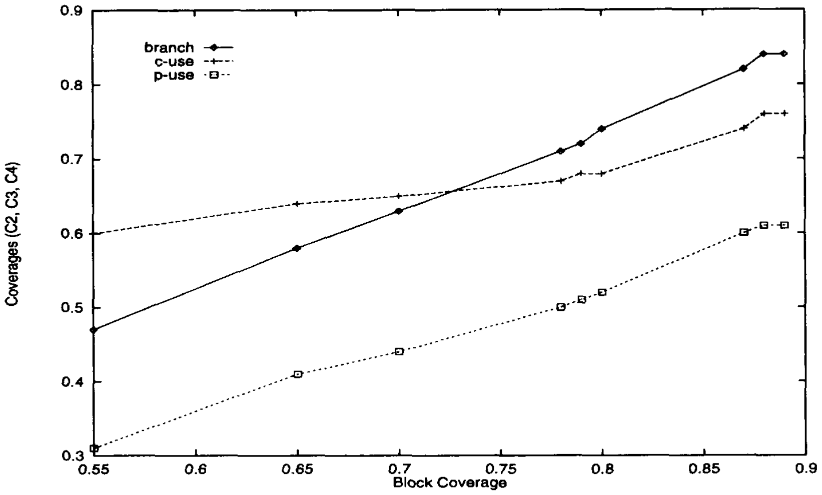

Figure 3: Scatter plot of C2, C3 and C4 against C1

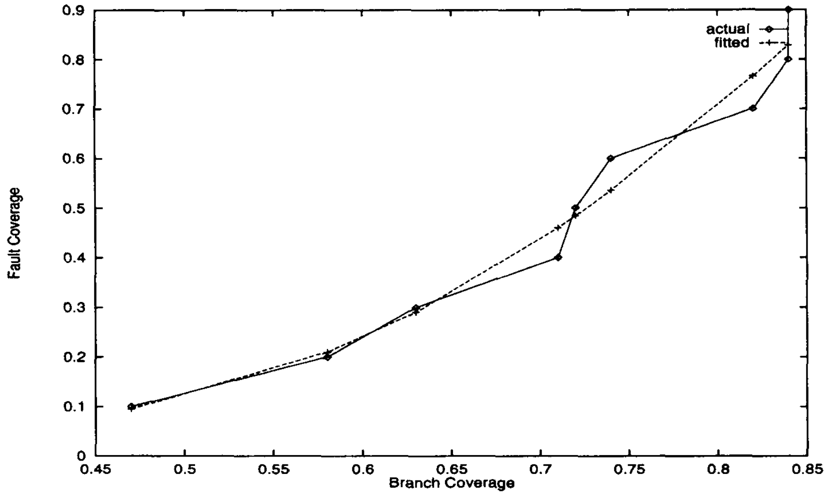

Figure 4: Actual and fitted (using Equation 11) values of defect coverage

$$b_0^i = \frac{K^i(0)N^i}{a^iT_LN^i} = \frac{K^i(0)}{a^iT_L} \qquad (12)$$

The linear execution time is given by the number of lines of code multiplied by the average execution time of each line. An empirical method to estimate the initial fault exposure ratio $K^0(0)$ has been suggested by Li and Malaiya [12]. Estimation of a^i remains an open problem. The second parameter is given by,

$$b^i = a^0T_s \qquad (13)$$

The single test execution time T_s depends on the program size and its structure. The product $b_0^ib_1^i$ then should be independent of the program size.

The parameters a_0^i, a_1^i, and a_2^i are defined in terms of b_0^i and b_1^i above. When this definition for a_0^i, a_1^i, and a_2^i is as an initial estimate for numerically fitting Equation 11, the initial estimate itself provides a least-square fit. If the initial estimates are significantly different, then the least square fit may yield somewhat different parameter values.

A-priori estimation of model parameters remains a partly unsolved problem. Currently we must rely on curve fitting based approaches.

8 Defect density and reliability

Since the failure intensity is proportional to the number of defects, we have [18, 19],

$$\lambda = \frac{K}{T_L}N$$

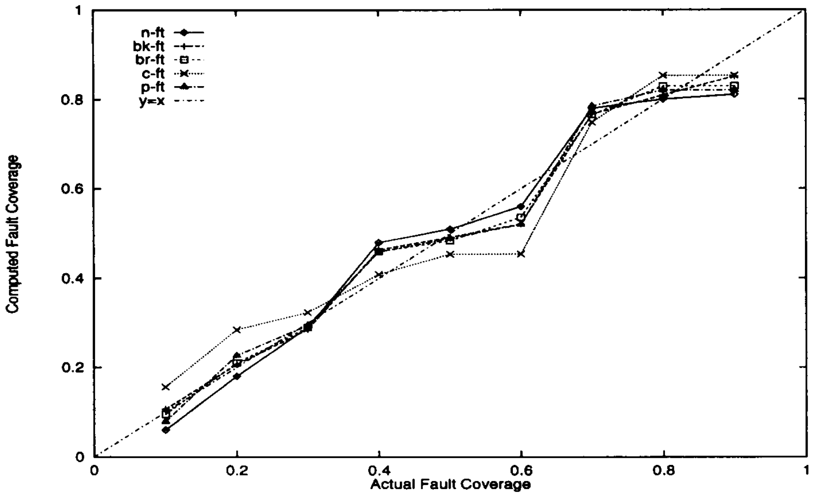

Figure 5: Actual defect coverage vs computed values

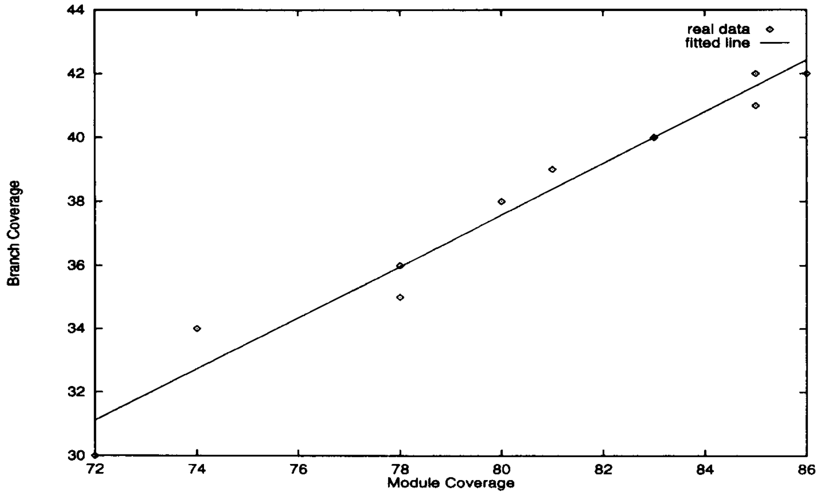

Figure 6: Scatter chart for module & branch coverages for an evolving program

Where K is the overall value of fault exposure ratio.

Let N_0 be the total number of faults initially present in the program and there is no new fault introduced during testing process. Then N can be computed as:

$$N = N_0(1 - C^0)$$

Substituting C^0 using Equation 11,

$$N = N_0(1 - a_0^i \ln[1 + a_1^i(exp(a_2^i C^i) - 1)])$$

Hence, the expected duration between successive failures can be obtained as

$$\frac{1}{\lambda} = \frac{T_L}{K} \frac{1}{N_0(1 - a_0^i \ln[1 + a_1^i(exp(a_2^i C^i) - 1)])} \quad (14)$$

Equation 14 can also be used for operational period using the appropriate value for the fault exposure ratio. Notice that K will depend on the operational profile used.

9 Conclusions and Discussions

We developed a modeling scheme that relates defect density to measurable coverage metrics. Defects have a detectability distribution like other coverage enumerables, and the same model may govern them. Two advantages of using a logarithmic model to describe test effort and enumerables covered are:

1. The logarithmic model is superior to other models for predicting the number of defects.

2. The logarithmic model can account for 100% coverage achieved in finite time. For high reliability applications, 100% block coverage night not be sufficient. A more strict coverage measure such as branch or p-use coverage can be used to further estimate the defect density.

The data sets used suggest that the model works well. The results are consistent with the analytical coverage inclusion relationships. Our model is simple and easily explained, and is thus suitable for industrial use.

The model given by Equation 11 can be used in two different ways. *Extrapolation* requires collecting data for part of the testing process, which is then used to estimate the applicable parameter values. These are used for making projections for planning the rest of the test effort. *A priori* parameter estimation requires empirical estimation of parameters even before testing begins. We have some observations on what factors control the parameter values. Further work is needed to fully develop these techniques and would include a careful study of enumerable exposure ratios.

As we show, any test coverage measure can be used to estimate the defect density, by using applicable parameter values. This raises an important question. Should several coverage measures be used or just one? Which individual measure (or selected set) provides the best estimate?

1. We need further studies to determine which coverage measure provide the best estimates of the number of defects. For DS2, we find that block coverage C^1 provides the best and c-uses coverage C^3 the worst fit. This result may be true for only specific data sets, or for specific coverage/defect density ranges. Since the different coverage measures can be strongly correlated, perhaps they may work equally well in many situations.

2. For very high reliability, we may need a "scale" that works in that region. If the requirements are such that 100% block coverage is not enough, branch or p-use coverage may be more appropriate. Branch coverage may be an adequate measure in many cases, since about 80% branch coverage often produces acceptable results [10]. However for testing of individual modules or for highly reliable software p-use may be better.

3. Researchers suggests the use of a weighted risk measure [1, 23, 21]. Weights are chosen on the basis of relative significance of each measure. As we find, structural coverage measures tend to be strongly correlated, and thus a weighted average may not provide more information than a single measure. We need to identify more independent measures. Other types of coverage measures like functional coverage may be suitable.

Our results can serve as a basis for further data collection and analysis. We need to examine the behavior at different fault densities, especially at very low defect densities (for highly reliable applications). We also need to validate the model for different testing strategies on the modeling scheme and the parameter values. In general, deterministic (coverage driven) is more efficient than true random testing. Testing using special values or use of equivalence partitioning can significantly compress the test time. Since test coverage measures provide direct sampling of the state of the software, we expect Equation 11 to hold because time is eliminated as a variable. Additional data will allow us to validate and refine our modelling scheme. In addition we need to develop schemes for evolving programs where new faults and other non-covered enumerables are being added.

10 Acknowledgement

We would like to thank Alberto Pasquini, Bob Horgan, Aditya Mathur and Peng Lu for discussions on this subject.

References

[1] H. Agrawal, J. Horgan, E. Krauser, S. London, "A testing-Based model and Risk Browser for C" *Proc. Int. Conf. Rel., Qual.Control & Risk Asses.*, Oct. 1993, pp 1-7.

[2] B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 1990, pp. 74-75, 161-171.

[3] J. Bieman and J. Schultz, "Estimating the Number of Test Cases Required to Satisfy the All-du-paths Testing Criterion," *Proc. ACM TAV3-SIGSOFT*, pp.179-186.

[4] J. Bieman and J. Schultz, "An Empirical Evaluation (and Specification) of the All-du-paths Testing Criterion," *Software Eng. J.*, Jan. 1992, pp. 43-51.

[5] M. Chen, J. Horgan, A. Mathur and V. Rego, "A time/structure based model for estimating software reliability," *SERC-TR-117-P*, Purdue University, Dec. 1992.

[6] S. Dalal, J. Horgan and J. Kettenring, "Reliable Software and Communications: Software Quality, Reliability and Safety," *Proc. 15th Int. Conf. Software Eng.*, May 1993, pp. 425-435

[7] J. Dunham, "Experiments in software reliability: Life Critical Applications," *IEEE Trans. Soft. Eng.*, Jan. 1986, pp. 110-123.

[8] P. Frankl and E. Wayuker, "An Applicable Family of Data Flow Testing Criteria," *IEEE Trans. Soft. Eng.*, Oct. 1988, pp. 1483-1498.

[9] P. Frankl and N. Weiss, "An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing," *IEEE Trans. Soft. Eng.*, Aug. 1993, pp. 774-787.

[10] R. Grady, *Practical Software Metrics for Project Management and Process improvement*, PTR Prentice-Hall, 1992, pp. 58-60.

[11] H. Hecht and P. Crane, "Rare Conditions and Their Effect on Software Failures," Proceedings of *Ann. Reliability and Maintainability Symp.*, pp. 334-337, Jan. 1994.

[12] N. Li and Y. Malaiya, "Fault Exposure Ratio and Reliability Estimation," *Proc. 3rd Workshop Issues Software Reliability*, Nov. 1993, pp. 6.3.1-6.3.18.

[13] N. Li and Y. Malaiya, "Enhancing Acuracy of Software reliability Prediction" *ISSRE'93*, pp. 71-79.

[14] M. Lyu, J. Horgan and S. London, "A Coverage Analysis Tool for the Effectiveness of Software Testing" ISSRE'93, pp. 25-34.

[15] Y. Malaiya and S. Yang, "The Coverage Problem for Random Testing," Proc. *Int. Test Conference*, pp. 237-242, Oct. 1984.

[16] Y. Malaiya, N. Karunanithi and P. Verma, "Predictability of Software Reliability Models," *IEEE Trans. Reliability*, Dec. 1992, pp. 539-546.

[17] J. Musa, A. Iannino, K. Okumoto, *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, 1987.

[18] Y. Malaiya, A. von Mayrhauser and P. Srimani, "The Nature of Fault Exposure Ratio," *Proc. IEEE Int. Symp. Soft. Rel. Eng.*, Oct. 1992, pp. 23-32.

[19] Y. Malaiya, A. von Mayrhauser and P. Srimani, "An Examination of Fault Exposure Ratio," *IEEE Trans. Software Eng.* Nov., 1993, pp. 1087-1094.

[20] S. Ntafos, "A Comparision of Some Structural Testing Strategies" *IEEE Trans. Software Eng.*, June 1988, pp.868-874.

[21] A. Neufelder, *Ensuring Software Reliability*, Marcel Dekker Inc., 1993, pp. 137-140.

[22] P. Piwowarski, M. Ohba and J. Caruso, "Coverage measurement experience during function test," *ICSE'93*, pp. 287-300

[23] R. Poston, "The Power of Simple Software Testing Metrics", *Software Testing Times*, Vol. 3, No. 1993.

[24] J. Ramsey and V. Basili, "Analyzing the Test Process Using Structural Coverage", *Proc. ICSE'85*, pp. 306-312.

[25] S. Seth, V. Agrawal and H. Farhat, "A Statistical Theory of Digital Circuit Testability," *IEEE Trans. Comp.*, Apr. 1990, pp. 582-586.

[26] M. Trachtenberg, "Why Failure Rates observe Zipf's Law in Operational Software," *IEEE Trans. Reliability*, Sept. 1992, pp. 386-389.

[27] J. Voas and K. Miller, "Improving the Software Development Process Using Testability Research," *ISSRE'92*, pp. 114-121.

[28] M. Vouk "Using Reliability Models During Testing With Non-operational Profiles," *Proc. 2nd Bellcore/Purdue workshop issues in Software Reliability Estimation*, Oct. 1992, pp. 103-111

[29] K. Wagnor, C. Chin and E. McCluskey, "Pseudorandom Testing," *IEEE Trans. Comp.*, Mar. 1987, pp. 332-343.

[30] E. Weyuker, "An Empirical Study of the Complexity of Data Flow Testing," *Proc. TAV2*, July 1988.

[31] E. Weyuker, "More Experience with Data Flow Testing", *IEEE Trans. Soft. Eng.*, Sept. 1993, pp. 912-919.