Estimating Defect Density Using Test Coverage

Yashwant K. Malaiya
Jason Denton
Computer Science Dept.
Colorado State University
Fort Collins, CO 80523
`malaiya|denton@cs.colostate.edu`

Technical Report CS-98-104

Estimating Defect Density Using Test Coverage

Yashwant K. Malaiya
Jason Denton
Computer Science Dept.
Colorado State University
Fort Collins, CO 80523
`malaiya|denton@cs.colostate.edu`

Abstract

Defect density is one of the most important factors that allow one to decide if a piece of software is ready to be released. In theory, one can find all the defects and count them, however it is impossible to find all the defects within any reasonable amount of time. Estimating defect density can become difficult for high reliability software, since the remaining defects can be extremely hard to test. Defect seeding will work only if the distribution of seeded defects is similar to the existing defects. One possible way is to apply the exponential SRGM and thus estimate the total number of defects present at the beginning of testing. Here we show the problems with this approach and present a new approach based on software test coverage. Software test coverage directly measures the thoroughness of testing avoiding the problem of variations of test effectiveness. Here we present interpretations of the parameters of the coverage- defect-density model presented by Malaiya et al. We apply this model to actual test data to project the residual defect density. The results show that this method results in estimates that are more stable than the existing methods. This method is easier to understand and the convergence to the estimate can be visually observed.

1 Introduction

Defect density is among the most important measures of software reliability. In a survey by Revision Labs, the participants, mostly quality assurance engineers and managers, were asked " What is the most important way that you measure quality?". Fifty-four percent of the participants mentioned a defect count based measure, total defects classified by severity, defects per KLOC (1000 lines of code) or defects per function point [rev97]. This suggests that the number of defects is often used as a major acceptance criterion. Leading edge software development organizations typically achieve a defect density of about 2.0 defects/KLOC [bin97]. The NASA Space Shuttle Avionics software with an estimated defect density of 0.1 defects /KLOC is regarded to be an example of what can be currently achieved by the best methods [hat97]. A low defect density can be quite expensive to achieve, the Space Shuttle code has been reported to have cost about $1,000 per line of code. The cost of fixing a defect later can be several orders of magnitude higher than during development, yet a program must be shipped by some deadline dictated by market considerations. This makes the estimation of the defect density a very important challenge.

One conceivable way of knowing the exact defect density of a program is to actually find all remaining defects. This is obviously infeasible for any commercial product. Even if there are resources available, it will take a prohibitive amount of time to find all bugs in a large program [but93]. Sampling based methods have been suggested for estimating the number of remaining defects. McConnell [mcc97] has given a method that involves using two independent testing activities, perhaps by two different teams of testers. If N_A is the number of defects found by team A, N_B is the number of defects found by team B and if the number of defects found by both is N_{AB}, then we can assume that

$$\frac{N_{total}}{N_A} = \frac{N_B}{N_{AB}} \tag{1}$$

which allows us to estimate the total number of defects as

$$N_{total} = \frac{N_B \times N_A}{N_{AB}} \tag{2}$$

The equation 2 is based on the assumption that team A will find the same fraction of all possible faults as the fraction of all faults found by team B. In other words it assumes that the faults *found* have the same testability as the faults *not found*. However, in actual practice, the faults not found represent faults that are harder to find [mal84]. Thus equation 2 is likely to yield an estimate of faults that are relatively easier to find, which will be less than the true number. Ebrahimi [ebr97] has suggested a sampling based method to estimate the number of defects during the inspection phase. Fault seeding is another sampling method that estimates the total number of faults based on the number of seeeded faults [mcc97]. Again seeded faults are likely to be defects with higher testability resulting in underestimation of the number of faults.

It is possible to estimate the defect density based on past experience using empirical models like the Rome Lab model [lak97] or the model proposed by Malaiya and Denton [mal97]. The estimates obtained by such models can be very useful for initial planning, however these models are not expected to be accurate enough to compare with methods involving actual test data.

Another possible way to estimate the number of faults is by using the exponential SRGM. It assumes that the failure intensity is given by

$$\lambda(t) = \beta_0^E \beta_1^E e^{-\beta_1^E t} \tag{3}$$

It can be shown that the parameters β_0^E and β_1^E depend on the system and the test process characteristics. Specifically β_0^E represents the total number of defects that would be eventually found. We can estimate the number of remaining defects by subtracting the number of defects found from the value of β_0^E obtained by fitting. We will evaluate this approach by comparing it with a new approach presented here.

An SRGM relates the number of defects found to the testing time spent. In actual practice, the defect finding rate will depend on the test effectiveness that can vary depending on the test input selection strategy. A software test coverage measure (like block coverage, branch coverage, P-use coverage etc.) directly measures the extent to which the software under test has been exercised. Thus we can expect that a suitably chosen test coverage measure can correlate better with the number of defects encountered. The relationship between test coverage and the number of defects found has been investigated by Piwowarski, Ohba and Caruso [poc93], Hutchins, Goradia and Ostrand [hfgo94], Malaiya et al [mali94], Lyu, Horgan and London [lyu93] and Chen, Lyu and Wong [che97].

In the next section, a model for defect density in terms of test coverage is introduced and its applicability is demonstrated using test data. Section 3 presents an interpretation of the model parameters and a two-parameter approximation of the model. In the next section we use the model to estimate the number of defects and compare the results with those obtained using the exponential SRGM. Finally we present some observations on this new approach.

2 Coverage model for Defect Density

Recently a model was presented by Malaiya et al that relates the density of residual defects with test coverage measures [mali94]. This model assumes that the logarithmic SRGM is applicable to the total number of defects found, as well as the number of test enumerables (e.g. branches or p-uses). Here we use the superscript 0 for defects and i = 1,2,.. for various test enumerables. Thus we can write for defect coverage $C^0(t)$

$$C^0(t) = \frac{\beta_0^0}{N_0^0} ln(1+\beta_1^0 t), \;\; C^0(t) \leq 1 \tag{4}$$

and enumerable i coverage

$$C^0(t) = \frac{\beta_0^i}{N_0^i} ln(1+\beta_1^i t), \;\; C^i(t) \leq 1 \tag{5}$$

where N_0^0 is the total initial number of defects and N^i is the total number of enumerables of type i in the program under test. Here $\beta_0^0, \beta_1^0, \beta_0^i, \beta_1^i$, are appropriate SRGM parameters.

We can eliminate time t from equation 4 and 5 to obtain

$$C^0(C^i) = a_0^i ln[1+a_1^i(e^{a_2^i C^i}-1)], \;\; C^i \leq 1 \tag{6}$$

where

$$a_0^i = \frac{\beta_0^0}{N_0^0}, \tag{7}$$

$$a_1^i = \frac{\beta_1^0}{\beta_1^i} \tag{8}$$

and

$$a_2^i = \frac{N^i}{\beta_0^i} \tag{9}$$

If we indicate the total expected number of defects found in time t by $\mu^0(t)$, we can write $C^0(t) = \frac{\mu^0(t)}{N_0^0}$. Hence from equation 6,

$$\mu^0(C^i) = a_3^i ln[1+a_1^i(e^{a_2^i c^i}-1)], \;\; C^i \leq 1 \tag{10}$$

where $a_3^i = a_0^i \dot{N}_0^0 = \beta_0^0$

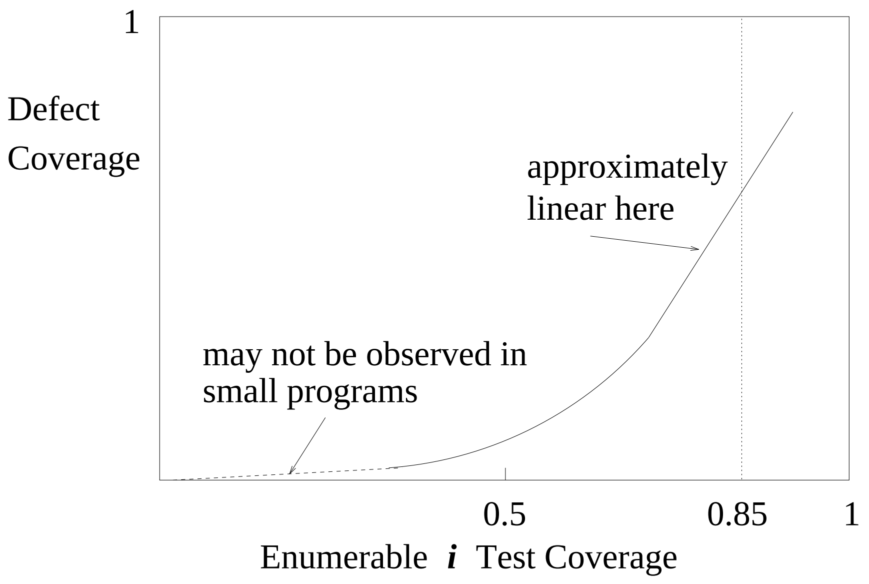

Figure 1: Defects vs. Test Coverage Model

Equation 10 can be used when the initial number of defects is not available. We must note that equations 6 and 10 are applicable only when C^i is less than or equal to one. Thus 100% branch coverage does not imply 100% defect coverage.

Figure 1 shows a plot illustrating the shape of the curve described by equation 6. At the beginning, defect coverage grows only slowly with test enumerable coverage. However, at higher test coverage, there is a linear relationship. The value around which the curve exhibits a knee has a significance as we will see below.

Applicability of this model is illustrated by the plots in figures 2, 3, and 4. This data was collected experimentally by Pasquini et al [pas97]. They tested a 6100 line C program by applying 20,000 test cases. The test coverage data was collected using the ATAC tool. Figure 2 shows a screen in ROBUST, an integrated software reliability evaluation tool [rob97] that has been developed at CSU. Further development of this tool is underway to include additional capabilities.

For the 20,000 tests, these were the coverage values obtained: block coverage : 82.31% of 2970 blocks, decision cover : 70.71% of 1171 decisions, and p-use coverage 61.51% of 2546 p-uses. This is to be expected since p-use coverage is the most rigorous coverage measure and block coverage is the least. Complete branch coverage guarantees complete block coverage, and complete p-use coverage guarantees complete decision coverage.

The plots suggest that 100% block coverage would uncover, 40 defects, 100% branch coverage would uncover 47 defects, whereas 100% p-use coverage would reveal 51 defects. We can expect that the actual total number of faults is slightly more than 51. In actual practice, it would be generally infeasible to achieve 100% p-use coverage. In Pasquini's experiment, it took an additional 18760 tests to increase p-use coverage from 66% to 67%. It is thus unlikely that more than 51 defects will be found with random testing within a reasonable period of time.

Also we note that the fitted model becomes very linear after the knee in the curve. Let us define C_k^i as the knee, where the linear part intersects the x-axis. For block, branch and p-use coverage it occurs at about 40%, 25% and 25% respectively. Below we see the significance of this value.

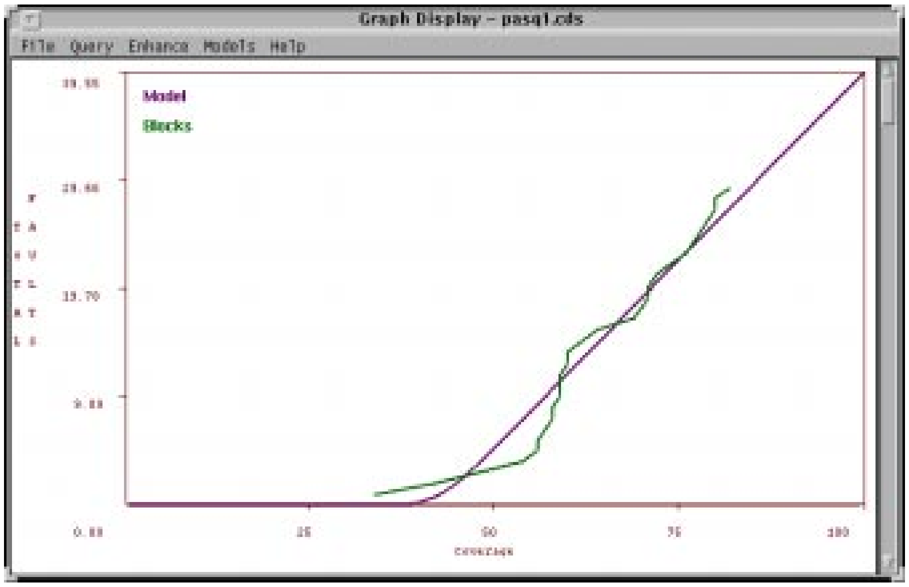

Figure 2: ROBUST: Block Coverage data (Pasquini et al.)

3 The Model Parameters: Interpretation

An interpretation of the three parameters in equation 10 is important for several reasons. Such an interpretation would provide valuable insight into the progress of testing to testers. Since the model is nonlinear and involves three parameters, accuracy of the results would be greatly enhanced if initial estimates using only static information like software size could be made.

Equations 7,8, and 9 relate the parameters of the coverage model to the logarithmic SRGM parameters. The parameters of the logarithmic model can be interpreted in two different ways, as shown by Malaiya and Denton [mal97]. Let us consider both approaches here.

3.1 Indirect Interpretation through the Exponential SRGM

This approach views the exponential model to be an approximation of the logarithmic model. Let us assume that at the end of test both models project the same total number of faults. Let us also assume that at the end the number of defects still not found is given by $\frac{N_0^0}{\alpha^0}$. We can then show that

$$\beta_0^0 = \frac{\alpha^0 - 1}{\alpha^0 ln(\alpha^0)} \beta_0^E \beta_1^0 = \frac{\alpha^0 - 1}{ln(\alpha^0)} \beta_1^E \tag{11}$$

where β_0^E and β_1^E are the parameters of the exponential models. Using the interpretation of the exponential model from [mus87], we have

$$\beta_0^0 = \frac{\alpha^0 - 1}{\alpha^0 ln(\alpha^0)} N_0^0 \tag{12}$$

and

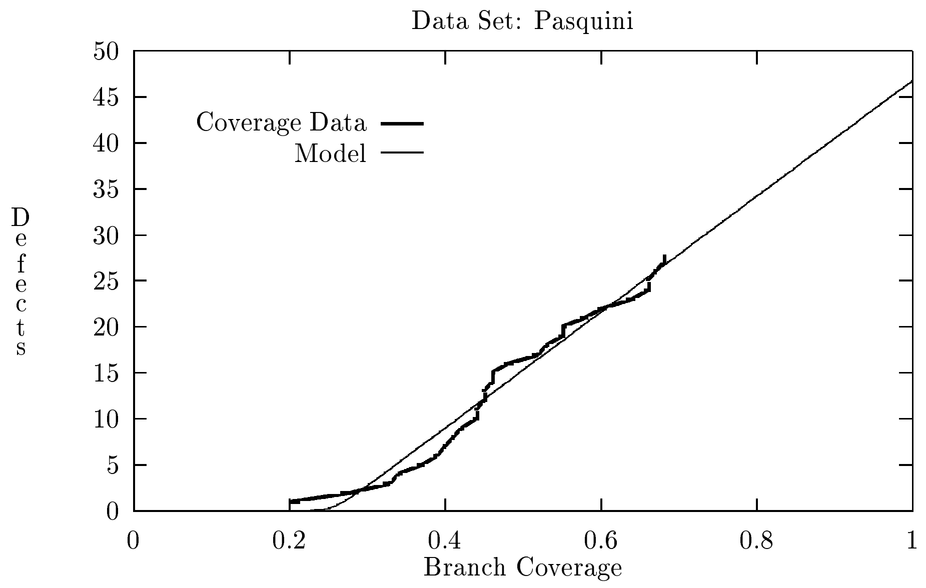

Figure 3: Defects vs. % Branch Coverage

$$\beta_1^0 = \frac{\alpha^0 - 1}{ln(\alpha^0)} \frac{\dot{K^0} r}{I_s Q} \tag{13}$$

where K^0 is the fault exposure ratio, I_s is the software source size, Q is the average number of object instructions generated per source instruction, and r is the object instruction execution rate of the computer used.

Using these equations, we can obtain from equations 7 - 9.

$$a_0^i = \frac{\alpha^0 - 1}{\alpha^0 ln(\alpha^0)} \tag{14}$$

$$a_1^i = \frac{\alpha^0 - 1 K^0}{ln(\alpha^0)} \frac{ln(\dot{a^i})}{a^i - 1} K^i \tag{15}$$

$$a_2^i = \frac{a^i ln(a^i)}{a^i - 1} \tag{16}$$

where

$$a^0 = \frac{\text{Total enumerables of type } i}{\text{Enumerables of type } i \text{ remaining uncovered}} \tag{17}$$

Generally we can expect that $a^i > a^0$ and $K^i > K^0$.

Example 1: Here we use Pasquini et al's data to obtain preliminary estimates of a_0^i, a_2^i, a_3^i. Let us assume that the total initial number of defects N_0^0 is 52. Let us also assume that the exponential SRGM and the Logarithmic SRGM track each other during testing until 51 of the 52 defects have been found. Then

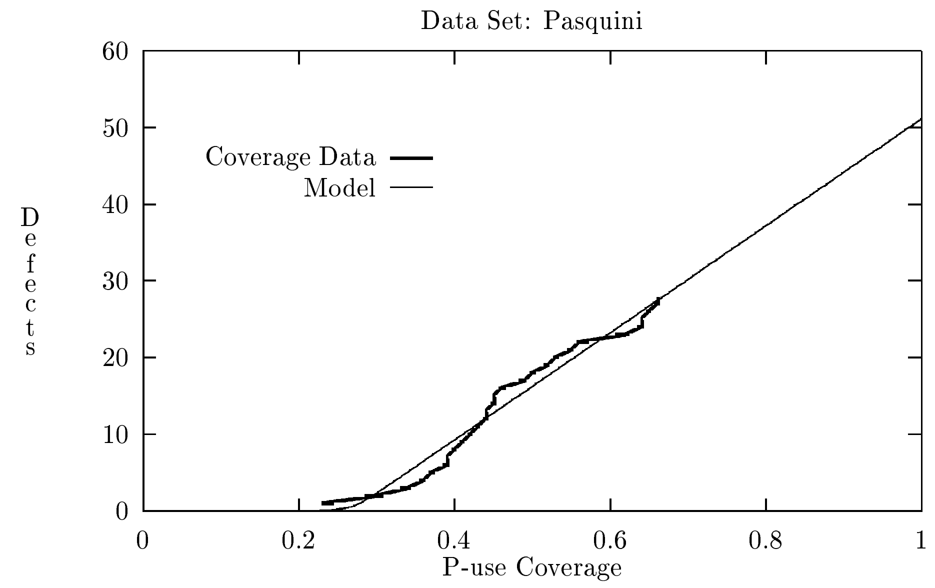

Figure 4: Defects vs. % P-use Coverage

$$\alpha^0 = \frac{52}{52-51} = 52.0 \tag{18}$$

Using this value, equation 14 gives

$$a_0^i = \frac{\alpha^0 - 1}{\alpha^0 ln \alpha^0} = 0.246 \tag{19}$$

Note that this value is not very sensitive to the value of N_0^0 assumed.

Let us use P-uses as the enumerable. Figure 4 suggests that 51 defects would be found with 98.53% P-Use coverage. Hence,

$$\alpha^i = \frac{100}{100-98.53} = 68.03 \tag{20}$$

Thus we have,

$$a_2^i = \frac{\alpha^i ln \alpha^i}{\alpha^i - 1} = 4.28 \tag{21}$$

we can also obtain

$$a_3^i = a_0^i N_0^0 = 0.248 \times 52 = 12.89 \tag{22}$$

Thus we can get preliminary estimates for two of the three parameters. The parameter a_1^i is harder to estimate, however we can see that it can be estimated using the estimates of the other two parameters.

3.2 Direct Interpretation through the Logarithmic SRGM

An interpretation of the logarithmic model parameters can be obtained by considering the variation in the fault exposure ratio [mvs93, mal97],

$$\beta_0 = I_s D_{min} \tag{23}$$

$$\beta_1 = \frac{K_{min} r}{Q I_s} e^{1 - \frac{D_0}{D_{min}}} \tag{24}$$

where D_O is the initial defect density, K_{min} is the minimum value of K and $Dmin$ is the defect density at which $K = K_{min}$. Both K_{min} and D_{min} are parameters that characterize the defect finding process, depending both on the software under test and the test strategy. Using these equations, we can write equations 4,5 and 6 as

$$a_0^0 = \frac{I_s D_{min}}{N_0^0} = \frac{D_{min}}{D_0^0} \tag{25}$$

$$a_1^0 = \frac{K_{min}^0 e^{D_0^0 / D_{min}^0}}{K_{min}^i e^{D_0^i / D_{min}^i}} \tag{26}$$

$$a_2^0 = \frac{N^i}{I_s D_{min}^i} = \frac{D^0}{D_{min}^i} \tag{27}$$

These three parameters are all independent of software size. Equation 27 suggests that a_0^i would have an inverse dependence on the initial defect density D_0^0. In actual practice, the testing strategy (and hence D_{min}^0) itself varies with D_0^0. Thus the dependence of D_0^0 variation on α_0^0 may be small.

Example 2: Here we will obtain preliminary estimates of the parameters using the direct interpretation. In [mal97] it has been observed that D_{min}^0 can often be estimated as $D_0/3$. This would give

$$a_0^i = \frac{D_{min}^0}{D_0^0} = \frac{D_0}{3} \frac{1}{D_0^0} = 0.33 \tag{28}$$

For an enumerable like P-uses, we do not yet know the relationship between D_0^i and D_{min}^i. If we assume that the minimum enumerable exposure ratio occurs at the same time as the minimum fault exposure ratio, we can estimate D_{min}^i as $D_0^i/3$. That yields,

$$a_2^i = \frac{D_0^i}{D_{min}^i} = \frac{D_0^i}{(D_0^i/3)} = 3.0 \tag{29}$$

Comparing with the values in Example 1, we see that both interpretations yield comparable preliminary estimates.

3.3 A Two parameter approximation

Figure 1, which is a direct plot of the model, and figures 2, 3, and 4 which plot actual data, suggest that at higher coverage values, a linear model can be a very good approximation.

We will obtain a linear model from equation 6. Let us assume that at higher value of C^i equation 6 can be simplified as

$$C^0(C^i) = a_0^i ln[a_1^i e^{a_2^i C^i}] = a_0^i ln(a_1^i) + a_0^i a_2^i C^i = A_0^i + A_1^i C^i, \ \ C^i > C_n^i \tag{30}$$

where

$$A_0^i = a_0^i ln(a_1^i) = \frac{\beta_0^0}{N_0^0} ln\left(\frac{\beta_1^0}{\beta_1^i}\right) \tag{31}$$

and

$$A_1^i = a_0^i a_1^i = \frac{\beta_0^0}{N_0^0}\frac{N^i}{\beta_0^i} \tag{32}$$

Note that this simplification is applicable only for values greater than C_n^i where the knee occurs. The experimental parameters values for this model can not be obtained until a clear linear behavior beyond the knee has been established. Assuming that the knee occurs where the linear part of the model intersects the x-axis, using equation 30, the knee is at

$$C_{knee}^i = -\frac{A_0^i}{A_1^i} \tag{33}$$

Here we can make a useful approximation. For a strict coverage measure, for $C^i = 1, C^0 \approx 1$. Hence from equation 30, we have

$$A_0^i + A_1^i \approx 1. \tag{34}$$

Replacing A_0^i by $1 - A_1^i$ in equation 33, and using 31, we can write,

$$C_{knee}^i = 1 - \frac{1}{a_0^i a_1^i} \tag{35}$$

Using the interpretation of the parameters through the Logarithmic model, this can be written as

$$C_{knee}^i = 1 - \left(\frac{D_{min}^i}{D_{min}^0 D_0^i}\right) D_0^0 \tag{36}$$

Where D_{min}^i, D_{min}^0, D_0^i are parameters and D_0^0 is the initial defect density. Thus for lower defect densities, the knee occurs at higher test coverage. This has a simple physical interpretation. If a program has been previously tested resulting in a lower defect density, it is likely that the enumerables with higher testability have already been exercised. This means that testing will start finding a significant number of additonal defects only after higher test coverage values are achieved.

The approximation 34 has another useful implication. Using equations 31 and 32, we can obtain,

$$a_1^i \approx e^{\frac{1-a_0^i a_2^i}{a_0^i}} \qquad (37)$$

which can be used to estimate a_1^i once a_0^i and a_2^i have been estimated.

4 Estimation of Defect Densities

The coverage model in Equations 6 and 10 provides us a new way to estimate the total number of defects. As we can see in Figures 2, 3 and 4, which use the data obtained by Pasquini et al., the data points follow the linear part of the model rather closely. Both the experimental data and the model suggest that the 100% coverage eventually achieved should uncover the number of faults as given in Table 1 below. The numbers in the last column have been rounded to nearest integer.

Table 1: Projected number of total defects with 100%coversage

Coverage measure	Total defects found	Coverage achieved	Defects expected with 100% coverage
Block Coverage	28	82%	40
Branch Coverage	28	70%	47
P-uses Coverage	28	67%	51
C-uses Coverage	28	74%	43

It should be noted that for this project 1240 tests revealed 28 faults. Another 18,760 tests did not find any additional faults, even though at least 5 more faults was known. The data suggests that the enumerables (blocks, branches etc.) not covered by the first 1240 tests were very hard to reach. They perhaps belong to sections of the code intended for handling special situations.

There is a subsumption relationship among blocks, branches and P-uses. Covering all P-uses assures covering all blocks and branches. Covering all branches assures coverage of all blocks. Thus among the three measures the P-use coverage measure is most strict. There is no coverage measure such that 100% coverage will assure detection of all the defects. Thus in the above table, the entry in the last column is a low estimate of the total number of defects actually present. Using a more strict coverage measure raises the low estimate closer to the actual value. Thus the estimate of 51 faults using P-use coverage should be closer to the actual number than the estimates provided by block coverage. Two coverage measures, DU-path coverage and all-path coverage are even more strict than P-use coverage, and may be suitable for cases where ultra-high reliability is required. However considering the fact that often even obtaining 100% branch coverage is infeasible, we are unlikely to detect more faults than what the P-use coverage data provides even with fairly rigorous testing. It should be noted that C-use coverage does not fit in the subsumption hierarchy and therefore it is hard to interpret the values obtained by using C-use coverage data.

Further application of this new method is illustrated by examining the data provided by Vouk [vou92]. These three data sets were obtained by testing three separate implementations of a sensor management program for an inertial navigation system. Each program is about five thousand lines of code. In the first program, 1196 tests found 9 defects. For the other two programs 796 test revealed 9 and 7 defects respectively. Figure 5 shows the plots of P-use coverage achieved versus the number of defects found. Table 2 shows the estimates for the total number of faults that would be found with 100% coverage.

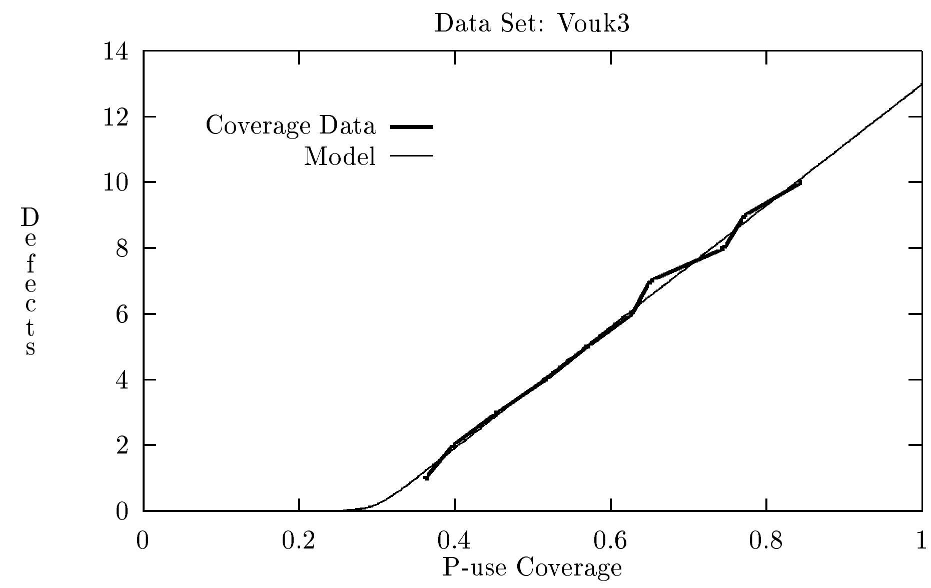

Figure 5: Defects vs. % P-use Coverage

Table 2: Projected number of total defects with 100%coversage (Vouk's data sets)

	Faults	Expected Faults		
Data Set	Found	Block	Branch	P-use
Vouk1	9.00	11.26	11.49	17.87
Vouk2	7.00	7.74	7.83	8.14
Vouk3	10.00	9.97	10.60	12.98

Table 2 again shows that the estimates obtained are consistent with the subsumption hierarchy.

4.1 Comparison with exponential model based approach

As mentioned before, it is possible to obtain an estimate of defect density from the exponential SRGM. The parameter β_0^E of this model represents the total number of faults in the system, and can be determined by fitting the available data to the model. Figure 6 shows the fitted value of β_0^E for Pasquini et al's data as testing progresses. Several things about this plot are worth noting.

First, towards the end of testing the exponential model consistently predicts that the total defects present is the same as the number of defects found. As figure 6 shows, defects are still being found after the point where the model predicts zero residual defects. This means that in the later stages of testing, the exponential model provides no useful information about the remaining defects. Second, the predictions made by the exponential model never stabilize. The estimate of β_0^E continues to rise as new defects are found, and can do so at a widely varying rate. If the estimates produced by the exponential model where accurate, then they should eventually begin to converge to some value, but this does not seem to be the case. Figure 8 shows similar results with data from Vouk.

Figure 7 shows the estimates for total defects made by our model as testing progresses, based on Pasquini et al's data. Like the exponential model, the predictions initially made by our model rise quickly. After about 20 test cases however, they begin to take on stable values, maintaining

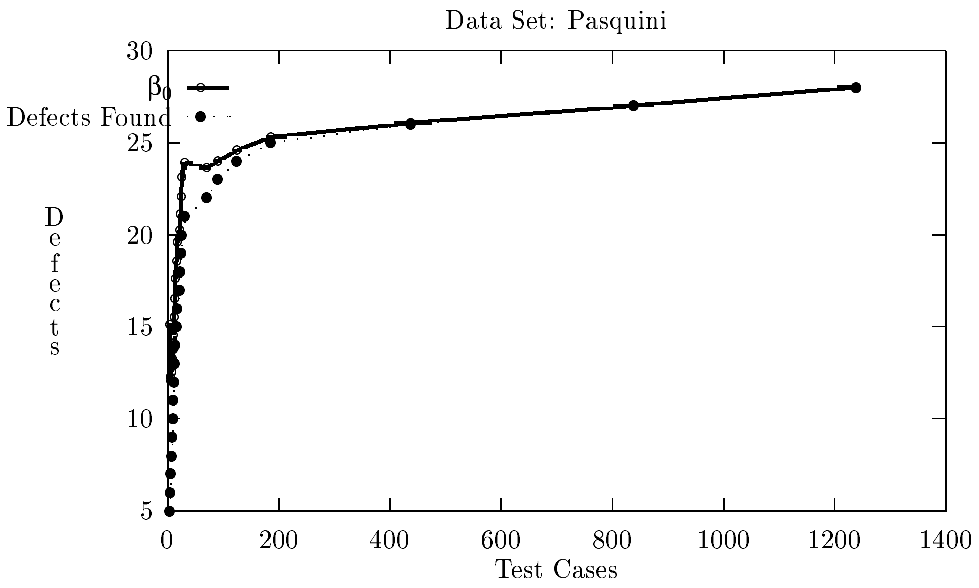

Figure 6: Estimated total defects using exponential model (Pasquini data)

consistent estimates of the total defects as more data comes in. Again, we see similar results with Vouk's data, as shown in figure 9. This stability is quite remarkable considering that the defect finding rate fluctuates considerablly.

It should be noted that if there is any unreachable or redundant code, our method will regard it as a part of the overall code. Presence of unreachable code can be avoided by using coverage tools and making sure that all modules and sections are entered during testing. The data sets used in this paper are for programs that are not evolving and thus the actual defect density is constant. The variation in the estimate of the total number of faults arise due to use of additional test data.

5 Conclusions

We have presented a model for defects found based on coverage, and shown that this model provides a very good description of actual data. Our method provides stable projections earlier in the development processes. The choice of coverage measure has an effect on the projections made. Results show that a strict coverage measure such as P-uses gives the most accurate results. We have provided interpretations of the model parameters, and shown how *a priori* information about the development process can be used to obtain estimates of the parameters in the model.

Defect density is an important measure of software reliability, figuring prominently in the reliability assessment of many quality assurance engineers and managers. Existing methods for estimating the defect can underestimate the number of defects because of a bias towards easily testable faults. The exponential model tends to generate unstable projections. This can lead to cost overruns or low reliability. Experimental data suggests that our method can provide more accurate estimates and provide developers with the information they require to make an accurate assessment of software reliability.

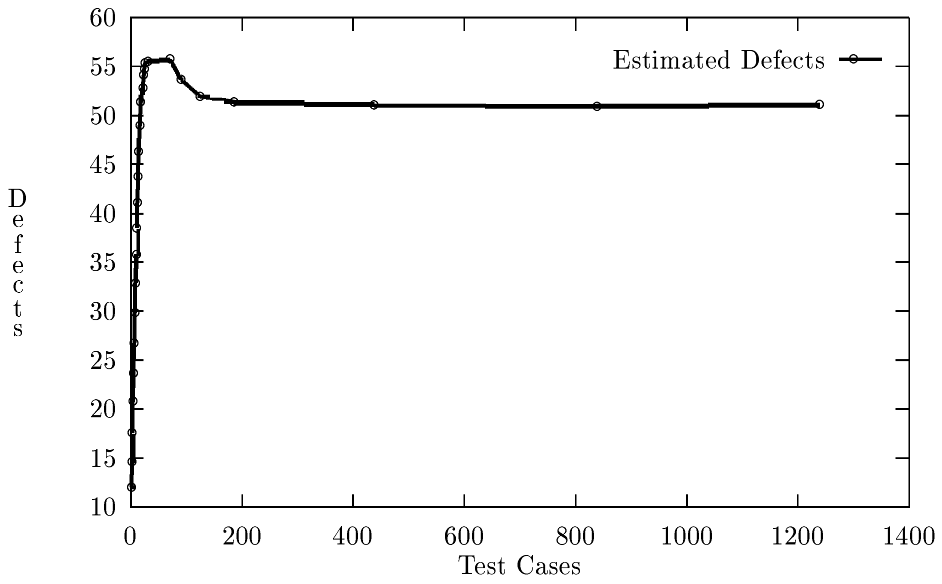

Figure 7: Estimated total defects with new approach (Pasquini data)

6 Acknowledgement

This work was supported in part by a BMDO funded project monitored by ONR. We would like to thank Mladen Vouk for providing us some of the data used here.

References

[bin97] Robert V. Binder, “Six Sigma: Hardware Si, Software No!”, http://www.rbsc.com/pages/sixsig.html, 1997.

[but93] R.W. Butler, and G.B. Finelli, “The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software. ”IEEE Transactions on Software Engineering, vol. 19, no. 1, Jan.1993, pp. 3-12.

[che97] Mei-Hwa Chen, Michael R. Lyu and W. Eric Wong, “An Empirical Study of the Correlation between Coverage and Reliability Estimation” IEEE Third Int. Symposium on Software Metrics, Mar. 25-26, Berlin, Germany, 1996.

[ebr97] Nader B. Ebrahimi, “On the Statistical Analysis of the Number of Errors Remaining in a Software Design Document after Inspection” IEEE Trans. on Software Engineering, Vol. 23, No. 8, August 1997, pp. 529-532.

[hat97] L. Hatton, “N-version Design Versus One Good Design” IEEE Software, Nov./Dec. 1997. pp. 71-76.

[hfgo94] M. Hutchings, T. Goradia and T. Ostrand, “Experiments on the effectiveness of data-flow and control-flow based test data adequacy criteria” International Conf. Software Engineering, 1994, pp. 191-200.

[lak97] P. Lakey, A. Neufelder, ”System and Software Reliability Assurance Notebook”, Rome Laboratory, 1997.

[li96] N. Li and Y.K. Malaiya, “Fault Exposure Ratio: Estimation and Applications” Proc. IEEE Int. Symp. Software Reliability Engineering 1996 pp. 372-381.

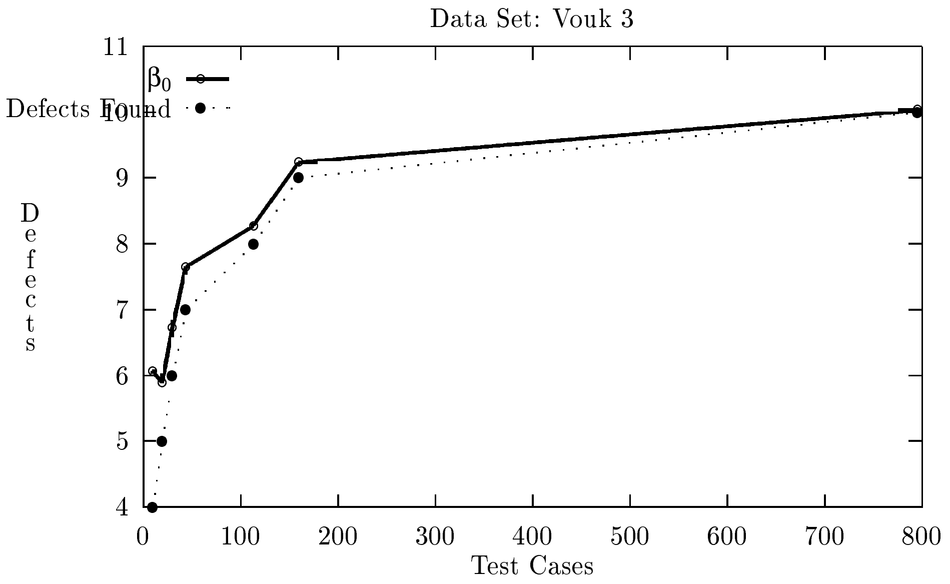

Figure 8: Estimated total defects using exponential model (Vouk data set 3)

[lyu93] M.R. Lyu, J.R. Horgan and S. London, "A Coverage Analysis Tool for the Effectiveness of Software Testing" IEEE Int. Symp. on Software Reliability Engineering, 1993, pp. 25-34.

[mal84] Y.K. Malaiya and S. Yang, "The Coverage Problem for Random Testing," *Proc. International Test Conference*, October 1984, pp. 237-245.

[mvs93] Y. K. Malaiya, A. von Mayrhauser and P. Srimani, "An examination of Fault Exposure Ratio," *IEEE Trans. Software Engineering*, Nov. 1993, pp. 1087-1094.

[mali94] Y.K. Malaiya, N. Li, J. Bieman, R. Karcich and B. Skibbe, "The Relationship between Test Coverage and Reliability" *Proc. Int. Symp. Software Reliability Engineering*, Nov. 1994, pp.186-195.

[mal97] Y. K. Malaiya and J. Denton, " What Do the Software Reliability Growth Model Parameters Represent?" *Int. Symp. on Software Reliability Engineering*, 1997. pp. 124-135.

[mcc97] Steve McConnell, "Gauging Software Readiness with Defect Tracking" IEEE Software, Vol. 14, No. 3, May / June 1997. pp.

[mus87] J.D. Musa, A Iannino, K. Okumoto, *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, 1987.

[pas97] A. Pasquini,A.N. Crespo and P. Matrella, "Sensitivity of Reliability Growth Models to Operational Profile Errors" IEEE Trans. Reliability, Dec. 1996, pp. 531-540.

[poc93] P. Piwowarski, M. Ohba and J. Caruso, "Coverage measurement experience during function test," *Proc. 15th Int. Conf. Software Engineering*, May 1993, pp. 287-300

[rob97] ROBUST, An integrated Software Reliability tool. Manual available at `http://www.cs.colostate.edu/testing/robust/manual.pdf`.

[rev97] Revision Labs' Survey (April 1 to August 2, 1997), http://www.revlabs.com/surresult.html Rev. 8/9/97.

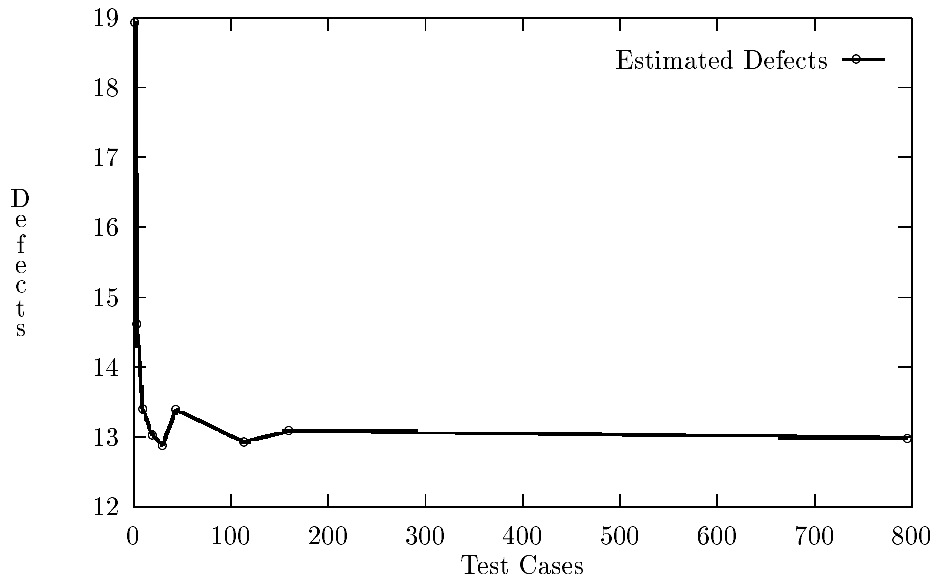

Figure 9: Estimated total defects with new approach (Vouk data set 3)

[vou92] M.A. Vouk "Using Reliability Models During Testing With Non-operational Profiles," *Proc. 2nd Bellcore/Purdue workshop on issues in Software Reliability Estimation,* Oct. 1992, pp. 103-111