Department of Computer Science

Empirical Estimation of Fault Exposure Ratio

Naixin Li and Yashwant K. Malaiya

Technical Report CS-93-113

August 20, 1993

Colorado State University

Empirical Estimation of Fault Exposure Ratio*

Naixin Li Yashwant K. Malaiya†
Computer Science Department
Colorado State University
Fort Collins, CO 80523
U.S.A.
Fax: (303) 491-6639
Office Phone: (303) 491-7031
Email: malaiya@cs.colostate.edu

Abstract

The fault exposure ratio K represents the average detectability of faults in a software. Knowing the value of K for a software system allows the use of the exponential model, and probably also the logarithmic model, even before software testing phase begins. It can also be used to enhance the software reliability growth models (SRGM) at the early stage of testing when the software failure data can only provides limited accuracy. This paper presents a model which relates the fault exposure ratio K to the initial defect density D_0. Empirical analysis indicates that this model can describe K fairly well.

*This research was partly supported by a SDIO/IST funded project monitored by ONR
†Contact person and presenting author

1 Introduction

The software reliability models are needed for measuring and projecting reliability. While many of the models are purely empirical, some of the models are based on some specific assumptions about the fault detection/removal process. The parameters of these models thus have some interpretations and thus possibly may be estimated using empirical relationships using static attributes. The two parameters of the exponential model are the easiest to explain. Using this model the expected number of faults $\mu(t)$ detected in a duration t may be expressed as

$$\mu(t) = \beta_0^E(1 - e^{-\beta_1^E t}) \tag{1}$$

Here β_0^E represents the total number of faults that would be eventually detected and β_1^E is the per fault hazard rate which is assumed to be constant for exponential model. The data collected by Musa [1] shows that the number of additional faults introduced during the debugging process is only about 5%. Thus β_0^E may be estimated as the initial number of faults. It has been observed [8] that in an organization, the defect density (measured in defects/thousand lines of code) at the beginning of the system test phase does not vary significantly and thus may be estimated with acceptable accuracy. Emperical methods to estimate defect density using programmer skill etc. have also been proposed [9, 10].

The estimation of the other parameter β_1^E is more complex. Musa et al have defined a parameter K, called **fault exposure ratio** which can be obtained by normalizing the per-fault hazard rate with respect to the linear execution frequency, which is the ratio of the instruction execution rate and the software size. For 13 software systems they found that K varies from 1.41×10^{-7} to 10.6×10^{-7}, with the average value equal to 4.2×10^{-7} failure/fault. Once we know the value of K, β_1^E can be estimated using,

$$\beta_1^E = \frac{K}{T_L} \tag{2}$$

where T_L is the linear execution time [1], given by $T_L = I_s Q_r \frac{1}{r}$; I_s is the number of source lines of code; Q_r is the average object instructions per source statement; r is the testing CPU instruction rate.

Musa et al [1] have speculated that K may depend on program structure in some way. However, they suspected that for large programs, the "structuredness" (as measured by

decision density) may average out and thus may not vary much with program size [1]. Musa has also argued that K should be independent of program size [2]. von Mayrhauser and Teresinki [11] have suggested that K may depend on static metrics like "loopiness" and "branchiness" of the program. However, because of lack of sufficient data, the results are not yet conclusive [12].

The logarithmic model, which is characterized by Equation 3 and Equation 4, has been empirically shown to be superior in general than other software reliability models by several researchers [14, 1, 6]. Malaiya et al [6] have shown using a number of diverse data sets that the superiority of the logarithmic model is statistically significant.

$$\mu(t) = \beta_0^L \ln(1 + \beta_1^L t) \tag{3}$$

$$\lambda(t) = \frac{\beta_0^L \beta_1^L}{1 + \beta_1^L t} \tag{4}$$

where $\lambda(t)$ is the failure intensity at time t. In paper [3] the parameter β_0^L was related to initial fault exposure ratio K_0 and β_1^L. In fact, if we can estimate K_0, the initial failure intensity λ_0 can be evaluated through,

$$\lambda_0 = \frac{K_0 N_0}{T_L} \tag{5}$$

Let $t = 0$ in Equation 4,

$$\lambda_0 = \beta_0^L \beta_1^L \tag{6}$$

Thus Equation 3 can be rewritten as,

$$\mu(t) = \frac{\lambda_0}{\beta_1^L} \ln(1 + \beta_1^L t) \tag{7}$$

So if we can estimate K_0, the logarithmic model will have only one unknown parameter β_1. Therefore it will be easier and probably more accurate to use at the early phase of testing.

The detectability of a fault is defined as the probability that the fault is detected by a randomly selected test input [3]. For truly random testing, faults with high detectability tends to be detected earlier in time, so K should in general decline with time [3]. However experiments with real reliability data indicates a reversal of this trend at low defect density areas. An explanation provided for this phenomenon was that at the late stage of testing phase testing becomes more and more directed. A model relating K with time, taking both

of these factors into account, as described by Equation 8 was proposed which satisfactorily characterized the variation of K with time t.

$$K(t) = \frac{g}{N(t)(1+at)} \tag{8}$$

where $N(t)$ is the number of faults present at time t; $g = K_0 N(0)$; a is a parameter depending on the "detectability profile" of the software [3].

From this model we can derive the well-known logarithmic software reliability growth model with the following interpretation for the parameters:

$$\beta_0^L = \frac{K_0 N_0}{a T_L} \tag{9}$$

$$\beta_1^L = a \tag{10}$$

A study of the factors affecting K is of considerable significance. If we can accurately model the behavior of K, there are two ways in which the debugging process can be better planned.

- When both parameters can be known *a priori*, the testing time needed to achieve target reliability can be calculated and hence, optimal resource allocation can be done even before testing begins.
- In early phases of testing, the failure intensity values observed contain considerable noise [7]. the use of reliability growth models in the early phases can sometimes result in grossly incorrect projection. The accuracy can be enhanced by using apriori parameter values in such cases.

In this work we are primary concerned with the relation between K and initial fault density we first derive the relation between K and defect density D, which will normalize the effects of factors like CPU instruction execution rate. Then we will apply this relation to a set of real data and see how well it fits the real behavior of software systems.

2 Variation of K with Fault Density

[3] derives an expression for K in term of testing time. Here we will obtain an expression for K in term of the fault density D. Taking logarithm on both sides of Equation 4, we get

$$\ln(\lambda(t)) = \ln(\beta_0^L \beta_1^L) - \ln(1 + \beta_1^L t) \tag{11}$$

Using Equation 3

$$\ln(\lambda(t)) = \ln(\beta_0^L \beta_1^L) - \frac{1}{\beta_0^L}\mu(t) \tag{12}$$

On the other hand, by definition,

$$D(t) = \frac{N(t)}{I_s} \tag{13}$$

and from [3],

$$\frac{dN(t)}{dt} = -\frac{K}{T_L}N(t) \tag{14}$$

Applying the following equation,

$$\lambda(t) = -\frac{dN(t)}{dt} \tag{15}$$

to Equation 14, we can obtain,

$$K(t) = T_L \frac{\lambda(t)}{N(t)} \tag{16}$$

Thus we have the following,

$$\ln(K(t)D(t)) = \ln(T_L \frac{\lambda(t)}{N(t)} \frac{N(t)}{I_s}) = \ln(T_L \frac{\lambda(t)}{I_s}) = \ln(\frac{I_s Q_r \frac{1}{r}}{I_s}\lambda(t)) \tag{17}$$

Reorganizing,

$$\ln(\lambda(t)) = \ln(K(t)D(t)) - \ln(\frac{Q_r}{r}) \tag{18}$$

Since $\mu(t)$ can also be expressed as,

$$\mu(t) = N_0 - N(t) = N_0 - I_s D(t) \tag{19}$$

Substituting Equation 18 and Equation 19 into Equation 12, we obtain,

$$\ln(K(t)D(t)) - \ln(\frac{Q_r}{r}) = \ln(\beta_0^L \beta_1^L) - \frac{1}{\beta_0^L}(N_0 - I_s D(t)) \tag{20}$$

Rearanging, we get,

$$\ln(K(t)D(t)) = \ln(\beta_0^L \beta_1^L) + \ln(\frac{Q_r}{r}) - \frac{N_0}{\beta_0^L} + \frac{I_s}{\beta_0^L}D(t) \tag{21}$$

Both defect density D and fault exposure ratio K should vary as testing proceeds. The exact process of defect detection and removal depends on the testing strategy, testing personnel, CPU speed, etc. To simplify this, we may assume that, K is a function of D only. Other factors will be manifested through variation in D. Hence the above equation can be abbreviated as:

$$\ln(KD) = \ln(\alpha_0) + \alpha_1 D \tag{22}$$

where the parameters α_0 and α_1 are given by,

$$\alpha_0 = \frac{\beta_0^L \beta_1^L Q_r}{r} e^{-\frac{N_0}{\beta_0^L}} \tag{23}$$

$$\alpha_1 = -\frac{I_s}{\beta_0^L} \tag{24}$$

This allow us to write K as,

$$K = \frac{\alpha_0}{D} e^{\alpha_1 D} \tag{25}$$

Equation 25 can also be derived from Equation 8 proposed in [3].

To validate the above assumption about K and D, the example from [3] is used here. We assume that a debugging process for a system with $N_0 = 200$ is exactly described by a logarithmic model with $\beta_0^L = 60$ and $\beta_1^L = 1$, we can calculate the values of K at different densities. The values are plotted in Figure 1 along with the fitted model of Equation 25. Figure 2 is a plot of $\ln(KD)$ against D. A perfect fit was shown on both cases as shown in the figures.

3 Estimation of K

Adams [13] noticed that software's failure rates in operational phase had similar distribution, which observes Zipf's law," the failure rate of a fault i is inversely proportional to a power of i, when faults are ranked by decreasing failure rate" [15]. Trachtenberg [15] proposed a software reliability model based on Zipf's law which fitted Adam's reliability data with an very high correlation coefficient.

If we regard the testing phase as a compressed form of operational use [1], then the failure rates during testing phase may also have similar distribution across software projects. Further we suspect that at the beginning of system testing phase, the detectability profiles [3] of software projects may have similar shapes in accordance with the Zipf's law.

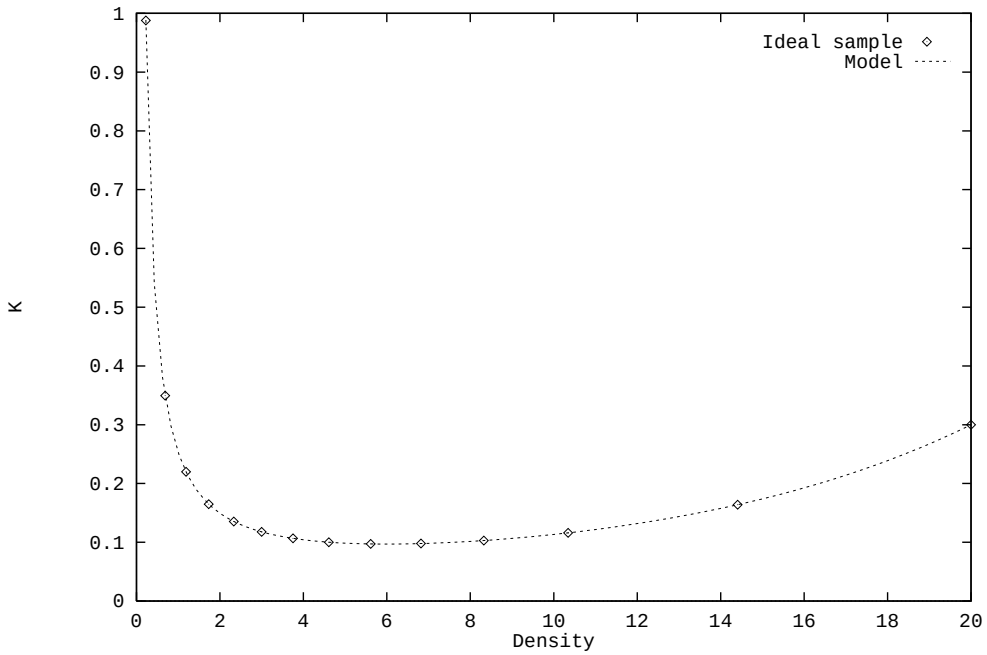

Figure 1: K against D for an ideal case

If the testing activity is conducted in the same way, K would be primarily determined by the detectability profile. In case of truely random testing, K would be an weighted average of the detectability of each individual fault [3]. We can assume that for a software system, the initial fault exposure ratio K_0 is determined mainly by the initial defect density D_0. If the above assumption about similar detectability profile shape for different software systems is valid, Equation 25 can be rewritten as,

$$K_0 = \frac{\alpha_0}{D_0} e^{\alpha_1 D_0} \tag{26}$$

Presently sufficient data is not available which will allow us to relate K_0 to D_0. Hence we cannot apply this model to any real data to evaluate the parameters α_0 and α_1. However, for the exponential model assumes that K does not change over time. Therefore, the overall value of K can be used for the exponential model as a constant. Table 1, obtained using [1], relates initial defect density and overall fault exposure ratio. Here 10% of the total initial defects are assumed to be present by the end of testing phase.

In Musa et al [1], the values of K were computed assuming the exponential model. According to the exponential model, from Equation 1,

$$N_0 - \mu(t) = \beta_0^E e^{-\beta_1^E t} \tag{27}$$

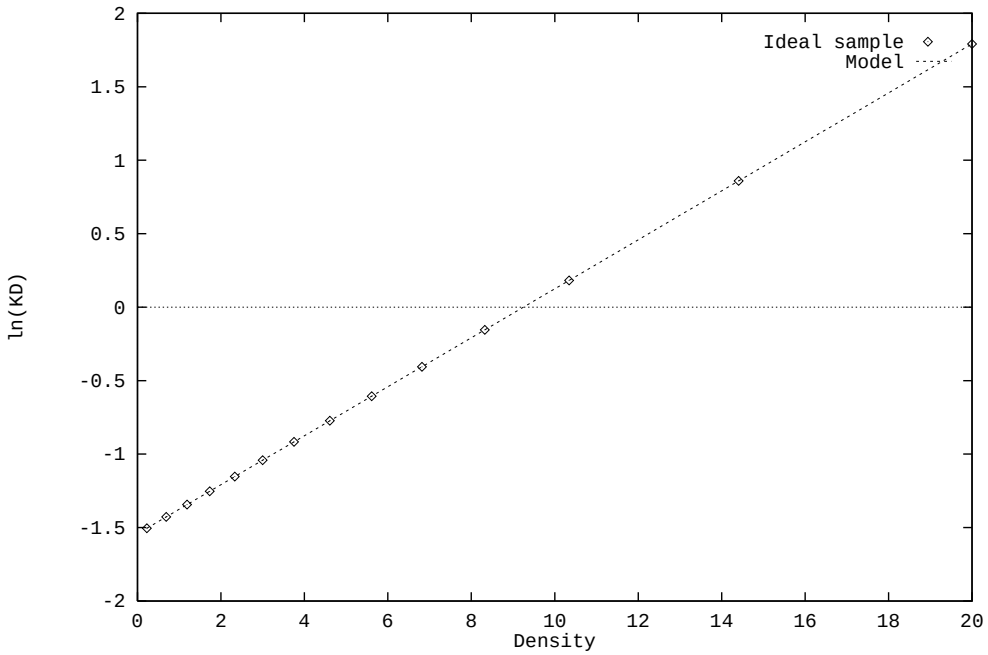

Figure 2: ln(KD) against D for an ideal case

or,

$$N(t) = \beta_0^E e^{-\beta_1^E t} \tag{28}$$

Since $N(t)/I_s = D(t)$, $N_0/I_s = D_0$, we have

$$D(t) = D_0 e^{-\beta_1^E t} \tag{29}$$

Let us assume that t varies from 0 to T such that $D(T) = D_0/10$, then

$$T = -\frac{1}{\beta_1^E}\ln(0.1) = \frac{2.303}{\beta_1^E} \tag{30}$$

Now the average value of $D(t)$ from 0 to T is given by

$$\hat{D} = \frac{1}{T}D_0 \int_0^T e^{-\beta_1^E t} dt = \frac{D_0}{\beta_1^E T}(1 - e^{-\beta_1^E T}) \tag{31}$$

Substituting Equation 30, we get

$$\hat{D} = \frac{D_0}{2.303}(1 - 0.1) = 0.39 D_0 \tag{32}$$

Since Equation 32 is a simple linear expression, D_0 may be used instead of $\hat{D}$ in an empirical analysis. The value of D_0 has been used in our regression.

Table 1: K vs. D_0 [1]
(K in units of 10^{-7})

Data	D_0	$\hat{D}$	K
T20	21.17	8.2563	6.5
T6	14.23	5.55	3.97
T1	6.96	2.71	1.87
T2	2.17	0.85	2.15
T3	1.8	0.7	4.11
T4	1.76	0.69	10.6
T19	0.68	0.27	4.54
T5	0.38	0.15	4.20
T16	0.36	0.14	3.03

Applying linear regression analysis to Equation 22 with the data shown in Table 1, we got the following estimation of parameters α_0 and α_1:

$$\alpha_0 = 3.089 \tag{33}$$

and,

$$\alpha_1 = 0.192 \tag{34}$$

The correlation coefficient value was 0.89 which was quite significant in indicating that the Equation 26 can satisfactorily describes the relation between K and D_0. If $\hat{D}$ was used in place of D_0, we would get $\alpha_0^{'} = 1.205$ and $\alpha_1^{'} = 0.492$ using Equations 26 and 32.

Figure 3 depicts the relation between K and D_0 for both the real data and the model with parameters evaluated from the real data. Figure 4 depicts $ln(K D_0)$ against D_0.

It was observed that the initial defect density for a software typically falls in the range of 3 to 20 defects per thousand lines of code. Using the above value of α_0 and α_1, we estimate the average value K to be in the range of [1.89..7.24], which is close to the range envisioned by Musa [1]. If more failure data are available, the value of α_0 and α_1 can be determined more exactly.

It should be noted that the defect density shown in Table 1 was calculated using the object instruction count. It might be possible to get even better correlation between K and D_0 if we consider only the projects using high-level languages, or only the projects which used assembly languages were used in a study.

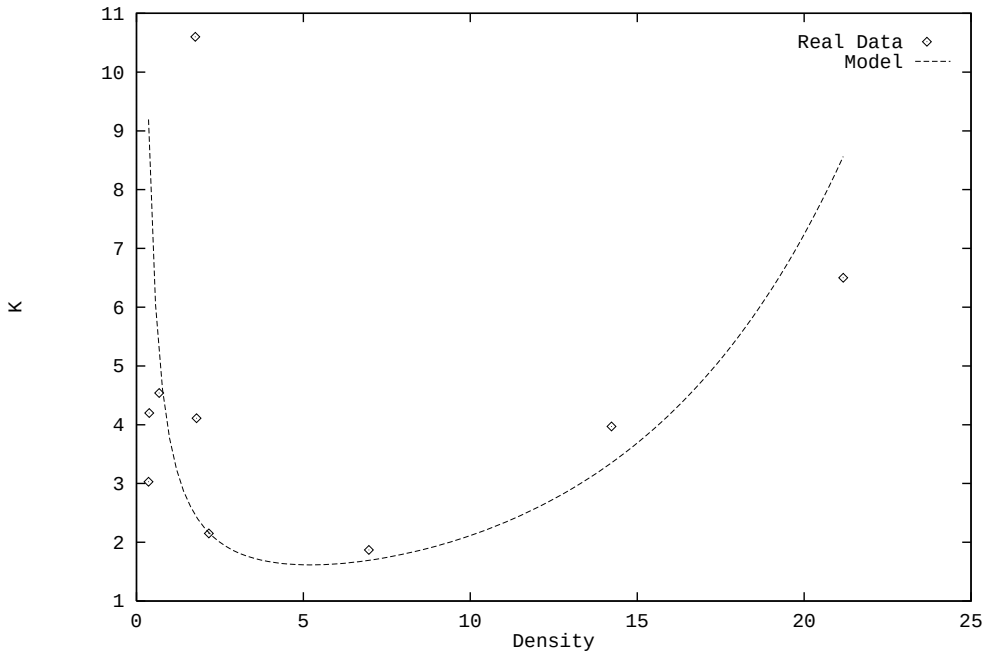

Figure 3: K against D from the real data

Although the parameters α_0 and α_1 evaluated here have been used for estimation of the average value of K, there is no reason that the model cannot be applied for estimation K_0 if there are real software reliability data available for evaluating the parameters of Equation 26.

4 An Example

In this section, we present an example to show how the results introduced in this paper may be used in practice. Assume there is a software system whose initial defect density is estimated to be $D_0 = 16$ defects per thousand lines of code. It has 50,000 lines of source code. The machine to be used for testing has MIPS rate of 16. The source to object instruction ratio is 4. Here we will calculate the total expected testing CPU time needed to achieve a failure intensity objective of 2×10^{-4}.

The total expect number of faults is

$$\beta_0^E = 50 \times 16 = 800 \tag{35}$$

The average fault exposure ratio is

$$\frac{\alpha_0}{16} e^{\alpha_1 \times 16} = \frac{3.089}{16} e^{0.192 \times 16} = 4.167 \times 10^{-7} \tag{36}$$

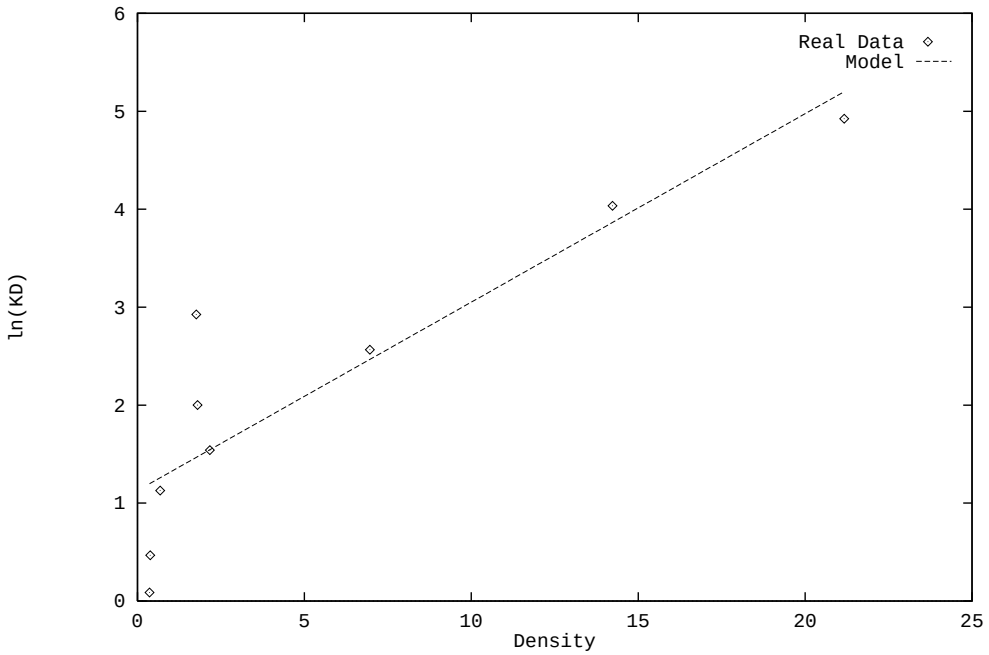

Figure 4: ln(KD) against D from real data

The linear execution time can be estimated as,

$$T_L = \frac{I_s Q_r}{r} = \frac{50000 \times 4}{16000000} = 0.0125 \tag{37}$$

The per fault hazard rate can be obtained as,

$$\beta_1^E = \frac{K}{T_L} = \frac{4.167 \times 10^{-7}}{0.0125} = 3.334 \times 10^{-5} \tag{38}$$

Thus we have the following exponential model describing the failure process during system testing,

$$\mu(t) = 800(1 - e^{-0.0000334t}) \tag{39}$$

and,

$$\lambda(t) = 0.0267e^{-0.00003334t} \tag{40}$$

where t is measured in CPU seconds. Solving this for the failure intensity objective, we get $t = 40.8$ CPU hours. By the time testing is terminated, there will be about 794 failures are expected.

Notice that this is obtained before real system testing starts. When enough actual failure data from system testing phase is available, one might consider to use real data and the logarithmic model to get a more accurate projection.

5 Concluding Remarks

We have presented an empirical model which allows us to estimate the fault exposure ratio and hence the parameter β_1^E of the exponential model. Since estimation of β_0^E can already be done satisfactorily, we can now use the exponential model before testing begins.

The fault exposure ratio K describes the effectiveness of the testing process. Besides the defect density, K may also depends on the testing strategy and perhaps the individual software structure. These may vary from project to project. However, within the same organization, these might have about the same effect for all the projects and thus K may be estimated directly from D. The estimation of K may be more accurate if the parameters α_0 and α_1 are obtained by fitting Equation 26 to the data from similar projects. When there is no previous data available within an organization, K_0 can be estimated using parameter values for α_0 and α_1 from a variety of projects from other organizations.

Just as size can be used to estimate the number of total fairly accurately, expected defects, the model here provides an estimate of K or K_0 using D_0. Further research is needed to identify and quantify the effect of other factors so that K or K_0 may be approximated more accurately, just as the frequency of specification changes, etc. can enhance the accuracy of estimating the total number of defects [9].

The model can be refined further when additional data is available. If there is data available to relate K_0 to D_0, then we can estimate K_0 and hence the parameter λ_0 of the logarithmic model (ref. Equation 7) at the beginning of system testing phase. Estimation of the other parameter β_1^L for the logarithmic model is yet an unsolved problem which is being investigated.

References

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

[2] J. D. Musa, Rationale for Fault Exposure Ratio K, ACM SIGSOFT Software Engineering News, July 1991, pp. 79.

[3] Y. K. Malaiya, A. von Mayrhauser and P. K. Srimani, The Nature of Fault Exposure Ratio, Proc. International Symposium on Software Reliability Engineering, October 1992, pp. 23-32.

[4] Y. K. Malaiya, Early Characterization of the Defect Removal Process, Proc. 9th Annual Software Reliability Symposium, May 1991, pp. 6.1-6.4.

[5] Y. K. Malaiya, A. von Mayrhauser and P.K.Srimani, The Constant Per Fault Hazard Rate Assumption, Proc. 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation, October, 1992, pp. 1-9.

[6] Y. K. Malaiya, N. Karunanithi and P. Verma, Predictability of Software Reliability Models, IEEE Trans. Reliability, December 1992, pp. 539-546.

[7] Y. K. Malaiya and P. Verma, Testability Profile Approach to Software Reliability, Advances in Reliability and Quality Control (Ed. M.H.Hamza), Acta Press, December 1988, pp. 67-71.

[8] G. A. Kruger, Validation and Further Application of Software Reliability Growth Models, Hewlett-Packard Journal, April 1989, pp. 75-79.

[9] M. Takahashi and Y. Kamayachi, An Emperical Study of a Model for Program Error Prediction, in Software Reliability Models, IEEE Computer Society, 1991. pp. 71-77.

[10] T. M. Khoshgoftar and J. C. Munson, The Live of Code Metric as a Predictor of Program Faults: a Critical Analysis", Proc. COMPSAC'90, pp. 408-413.

[11] A. von Mayrhauser and J. A. Teresinki, The Effects of Static Code Metrics on Dynamic Software Reliability Models, Proc. of Symposium on Software Reliability Engineering, April, 1990, pp. 19.1-19.13.

[12] J. M. Keables, Program Structure and Dynamic Models in Software Reliability: Investigation in a Simulated Environment, Ph.D Dissertation, Computer Science Dept., Illinois Institute of Technology, 1991.

[13] E. N. Adams, Optimizing Preventive Service of Software Products, IBM Journal of Research and Development, vol. 28, no. 1, January 1984, pp.2-14.

[14] W. H. Farr, A survey of Software Reliability Modeling and Estimation, Naval Surface Weapons Center, TR 82-171, Sept. 1983.

[15] M. Trachtenberg, Why Failure Rates Observe Zipf's Law in Operational Software, IEEE Trans. Reliability, vol. 41, no. 3, September 1992, pp. 386-389.