Computer Science
Technical Report

Colorado State University

What Do the Software Reliability Growth Model Parameters Represent? *

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

Technical Report CS-97-115

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

*This research was supported by a BMDO funded project monitored by ONR

What Do the Software Reliability Growth Model Parameters Represent? *

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

ABSTRACT

Here we investigate the underlying basis connecting the the software reliability growth models to the software testing and debugging process. This is important for several reasons. First, if the parameters have an interpretation, then they constitute a metric for the software test process and the software under test. Secondly, it may be possible to estimate the parameters even before testing begins. These *a priori* values can serve as a check for the values computed at the beginning of testing, when the test-data is dominated by short term noise. They can also serve as initial estimates when iterative computations are used.

Among the two-parameter models, the exponential model is characterized by its simplicity. Both its parameters have a simple interpretation. However, in some studies it has been found that the logarithmic poisson model has superior predictive capability. Here we present a new interpretation for the logarithmic model parameters. The problem of a priori parameter estimation is considered using actual data available. Use of the results obtained is illustrated using examples. Variability of the parameters with the testing process is examined.

1 Introduction

We can have two views of a software reliability growth model (SRGM). It can be regarded to be a mathematical expression which fits the experimental data. It may be obtained simply by observing the overall trend of reliability growth. However some of the models can be obtained analytically by making some assumptions about the software testing and debugging process. Some of these assumptions are simply to keep the analysis tractable. Other are more fundamental in nature and constitute modeling of the testing and debugging process itself.

An analytically obtained model has the advantage that its parameters have specific interpretations in terms of the testing process. An understanding of the underlying meaning of the parameters gives us a valuable insight into the process.

1. If we know how a parameter arises, we can estimate it even before testing begins. Such *a priori* values when estimated using past experience, can be used to do preliminary planning and resource allocation before testing begins [14].

*This research was supported by a BMDO funded project monitored by ONR

2. The experience with use of SRGMs suggests that in the beginning of testing, the initial test data yields very unstable parameter values and sometimes the parameter values obtained can be illegal in terms of the model. In such a situation, values estimated using static information can serve as a check. They can also be used to stabilize the projections adding to the information obtained by the dynamic defect detection data.

3. Sometimes iterative techniques are used to estimate the parameter values. The values obtained can depend on the initial estimates that are required by numerical computation. Use of *a priori* values as the initial estimate would initiate the search in a region closer to the values sought.

4. Parameters that have an interpretation characterize the testing and debuging process quantitatively. Their values can give us an insight into the process. They may help answer the questions about how the inherent defect density can be reduced or how testing can be made more efficient.

This paper examines the parameters of the exponential and the logarithmic models. We present a new model for estimating the software defect density. A new interpretation for the parameters of the logarithmic model is presented. Techniques for estimation of parameters are presented.

The quantitative process characteristic values used in this paper are taken from the data reported by researchers. The values depend on the process used and may be different for different process. Thus the models presented here should be recalibrated using the prior experience in a specific organization using a specific process. Similar methods have been in use for projecting hardware reliability measures where they have been found to be very useful even though the results are only approximate.

The next section analytically presents the interpretations of the parameters of the two models. Section 3 discusses estimation of parameters. Some observations on parameter variations are presented next followed by the conclusions.

2 Exponential and Logarithmic SRGMs

In this paper we will consider two two-parameter models. The Exponential model, in the formulation used here is also termed *Musa's basic execution model* [18]. It is given by

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

where $\mu(t)$ is the mean value function and β_0^E and β_1^E are the two model parameters.

Farr mentions that this model has had the widest distribution among the software reliability models [4]. Musa [18] states that the basic execution model generally appears to be superior in capability and applicability to other published models. Some of the other models are similar to this model.

The Logarithmic model is other model considered here. It is also termed *Musa-Okumoto logarithmic poisson Model*. It is given by

$$\mu(t) = \beta_0^L \ln(1 + \beta_1^L t) \tag{2}$$

where β_0^L and β_1^L are the two model parameters.

Farr states that the logarithmic model is one of the model that has been extensively applied [4]. This is one of the selected models in the AIAA Recommended Practice Standard. Musa [18] writes that the logarithmic model is superior in predictive validity compared with the exponential model. In a study using 18 data sets from diverse projects, Malaiya et al. evaluated the prediction accuracy of five two-parameter models [15]. They found that the logarithmic model has the best overall prediction capability. Using ANOVA, they found that this superiority is statistically significant.

All software reliability growth models (SRGMs) are approximations of the real testing process, thus none of the models can be regarded to be perfect. However these two models possess simplicity and have been found to be applicable for a variety of software projects. Thus these two models have been chosen for this study.

2.1 Derivation of the Exponential model

Here we give a derivation of the exponential model that gives its relationship with the test process. Let $N(t)$ be the expected number of defects present in the system at time t. Let T_s be the average time needed for a single execution, which is very small compared with the overall testing duration. Let k_s be the fraction of existing faults exposed during a single execution. Then

$$\frac{dN(t)}{dt}T_s = -k_s N(t)$$

It would be convenient to replace T_s with something which can be easily estimated. Let T_L be the *linear execution time* [18] which is defined as the total time needed if each instruction in the program was executed once and only once. It is given by

$$T_L = I_. Q_x \frac{1}{r}$$

where I_s is the number of source statements, Q_x is the number of object (machine level) instructions per source instructions and r is the object instruction execution rate of the computer being used.

Let us define a new parameter

$$K = k_s \frac{T_L}{T_s}$$

where the ratio $\frac{T_L}{T_s}$ will depend on the program structure.

Using this, equation 2.1 can be rewritten as

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

The per-fault hazard rate as given in equation 2.1 is K/T_L. Thus K, termed *fault exposure ratio* [18] directly controls the efficiency of the testing process. If we assume that K is time invariant, then the above equation has the following solution:

$$N(t) = N_0 e^{-\frac{K}{T_L} t}$$

Where N_0 is the initial number of defects. This may be expressed in a more familiar form as follows:

$$N_0 - N(t) = N_0(1 - e^{-t\frac{K}{T_L}})$$

The left side of this equation corresponds to $\mu(t)$, as given by equation 2. Thus the parameters β_0 and β_1 have the following interpretations:

$$\beta_0^E = N_0, \quad and \quad \beta_1^E = \frac{K}{T_L} \tag{3}$$

Experimental data suggests that K actually varies during testing [16]. We will denote the constant equivalent as determined by the application of the exponential model by $\hat{K}$.

2.2 Implications of the Logarithmic model

The logarithmic model has been found to have very good predictive capability in many cases. However to derive it from basic considerations would require one to make some assumptions as done in references [18], [17] and [16]. If the logarithmic model does describe the test process, that would imply that the fault exposure ratio is variable. We can assume that this variation depends on the test process phase which is given by the density of defect present at a time during testing [12].

Rearranging equation 2 for the mean value function $\mu(t)$, we can write,

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

Also,

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

Substituting for $(1 + \beta_1^L t)$ from equation 4

$$\lambda(t) = \beta_0^L \beta_1^L e^{-\frac{\mu(t)}{\beta_0^L}} = \beta_0^L \beta_1^L e^{-\frac{N_0 - I_s D(t)}{\beta_0^L}} \tag{5}$$

Where $D(t)$ is the defect density at time t. The fault exposure ratio is given by

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

Using equation 5 to substitute for $\lambda(t)$, we get

$$K(t) = \frac{T_L}{I_s D}\beta_0^L \beta_1^L e^{-\frac{N_0 - I_s D(t)}{\beta_0^L}} \quad (6)$$
$$= \left(\frac{T_L}{I_s D}\beta_0^L \beta_1^L e^{-\frac{N_0}{\beta_0^L}}\right) e^{-\frac{I_s D(T)}{\beta_0^L}}$$

We can rewrite this as

$$K = \frac{\alpha_0}{D} e^{\alpha_1 D} \quad (7)$$

where the parameters α_0 and α_1 are given by,

$$\alpha_0 = \frac{\beta_0^L \beta_1^L Q_x}{r} e^{-\frac{N_0}{\beta_0^L}} \quad (8)$$

$$\alpha_1 = \frac{I_s}{\beta_0^L} \quad (9)$$

The equations 8 and 9 are used in the next section to present a new interpretation for the logarithmic model.

3 Physical Interpretation of the Parameters

An interpretation of the parameters for the exponential model is quite straightforward. As $t \to \infty$, according to equation 1, $\mu(t) \to \beta_0^E$. Musa states that during debugging only about 5% new faults are introduced. Thus β_0^E is slightly greater than the initial number of faults, and can be taken to represent the total number of faults that will be encountered. The parameter β_1^E is the time scale factor, or the per fault hazard rate, as given by equation 3.

A greater challenge is posed by the logarithmic model parameters. Here we present a new interpretation based on the analysis presented in sec 2.2. From equation 8 we can write

$$\beta_0^L = \frac{I_s}{\alpha_1}$$

Substituting this in (16) and solving for β_1^L, we get

$$\beta_1^L = \frac{\alpha_0 r \alpha_1}{Q_r I_s} e^{\frac{N_0 \alpha_1}{I_s}}$$

Let us now determine the meaning of α_0 and α_1, in terms of the test process. Fig. 1 gives the variation of the fault exposure ratio K in terms of defect density. Let us denote by D_{min} the density at which K_{min}, the minimum value of K, occurs. Taking a derivative of K with respect to D using equation 7 and equating it to zero, we get

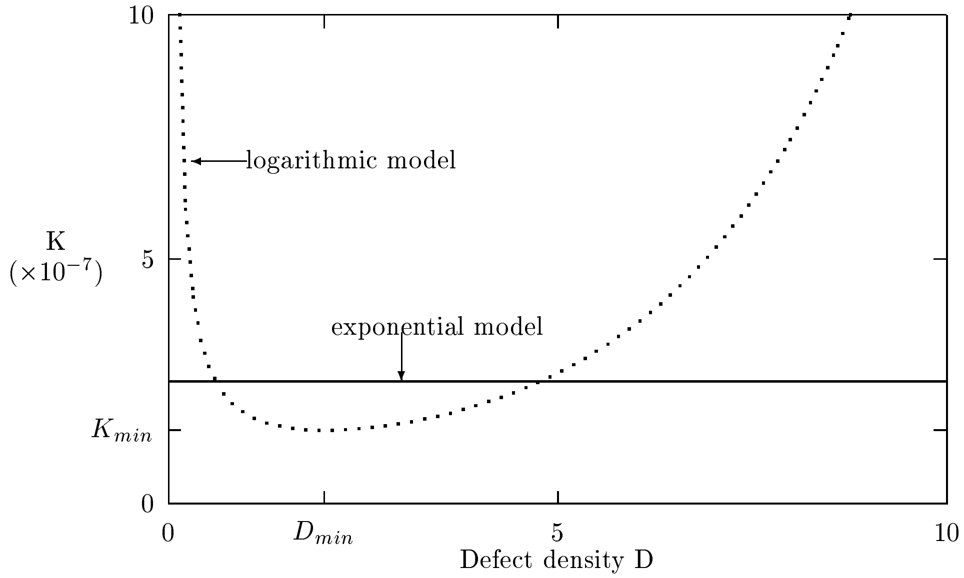

Figure 1: Variation of Fault Exposure Ratio with defect density

$$-\frac{\alpha_0}{D^2}e^{\alpha_1 D} + \frac{\alpha_0}{D}e^{\alpha_1 D}\alpha_1 = 0$$

which yields

$$D_{min} = \frac{1}{\alpha_1}$$

and the corresponding value of K is given by

$$K_{min} = \frac{\alpha_0 e}{D_{min}}$$

Thus both α_0 and α_1 depend on the test process,

$$\alpha_0 = \frac{K_{min}D_{min}}{e}, \quad and \quad \alpha_1 = \frac{1}{D_{min}} \tag{10}$$

Using equation 10, we obtain this interpretation of the logarithmic model parameters.

$$\beta_0^L = I_s D_{min} \tag{11}$$

$$\beta_1^L = \frac{K_{min} r}{Q_x I_s e} e^{\frac{D_0}{D_{min}}} \tag{12}$$

Equation 11 states that β_0^L is proportional to the software size and is controlled by how test effectiveness varies with defect density. The parameter β_1 depends on K_{min}, the minimum value of the fault exposure ratio. It is also dependent on the ratio $\frac{D_0}{D_{min}}$

It should be noted that β_0^E and β_0^L, and β_1^E and β_1^L have the same dimensions. The Table 1 below compares the interpretations of the parameters of the two models compared here.

Parameter	Dimension	Exponential	Logarithmic
Value Scale	Defects	$\beta_0^E \approx N_0 = D_0 I_s$	$\beta_0^L = D_{min} I_s$
Time Scale	Per unit time	$\beta_1^E = \frac{\hat{K}}{T_L}$	$\beta_1^L = \frac{K_{min}}{T_L} e^{\frac{D_0 - D_{min}}{D_{min}}}$

Table 1: Comparison of model parameter interpretations

3.1 Exponential Model Parameters:

Because the exponential model parameters are explained in a simpler way, the problem of a priori estimation of its parameters is also easier. Assuming the number of new faults introduced during the debugging process is small, β_0^E can be taken to be approximately equal to the initial number of defects, N_0. It has been observed that for a specific development environment for the same software development team, the defect density encountered is about the same, for the same development/testing phase. This allows the initial defect density to be estimated with reasonable confidence.

Here we present a *factor multiplicative* model to estimate the initial defect density and hence N_0. A factor multiplicative model assumes that the quantity to be estimated is influenced by several independent causes and the effect of each cause can be suitably modeled by a multiplicative factor. Such models have also been used to estimate the hardware failure rates. Several *linear additive* models for estimating the number of defects have also been proposed, they have the disadvantage that they can project zero or negative number of defects.

The models by Agresti and Evanco [2], Rome Lab [21] and THAAD [6] are factor multiplicative like our model. A preliminary version of our model [13] is being implemented in the ROBUST software reliability tool [10]. Our model, presented below, has the following advantages:

1. It can be used when only incomplete or partial information is available. The default value of a multiplicative factor is one, which corresponds to the average case.
2. It takes into account the phase dependence as suggested by Gaffney [5]
3. It can be recalibrated by choosing a suitable constant of proportionality and be refined by using a better model for each factor, when additional data is available.

The model is given by

$$D = C.F_{ph}.F_{pt}.F_m.F_s \tag{13}$$

where the four factors are the *phase factor* F_{ph}, modeling dependence on software test phase, the *programming team factor* F_{pt} taking in to account the capabilities and experience of programmers in the team, the *maturity factor* F_m depending on the maturity of the software development process and the *structure factor* F_s, depending on the structure of the software under development. The constant of proportionality C represents the defect density per thousand source lines of code (KSLOC). Here are the preliminary sub-models for each factor.

3.1.1 Phase Factor (F_{ph})

The number of defects at the beginning of different test phases is different. Gaffney [5] has proposed a phase based model that uses the Rayleigh curve. Here we present a simpler model using actual data reported by Musa et al. [18] and the error profile presented by Piwowarski et al. [20]. In Table 2 we take the default value of one to represent the beginning of the system test phase. With respect to this, the first two columns of Table 2 represent the multipliers suggested by the numbers given by Musa et al. and Piwowarski et al.. The third column presents the multipliers assumed by our model.

At the beginning of phase	Multiplier		
	Musa et al.	Piwowarski et al.	Our Model
Unit Testing	3.28	5	4
Subsystem Testing	Insufficient data	2.5	2.5
System testing	1	1	1 (default)
Operation Testing	0.25	0.45	0.35

Table 2: Phase Factor (F_{ph})

3.1.2 The Programming Team Factor (F_{pt})

The defect density varies significantly due to the coding and debugging capabilities of the individuals involved [22] [23]. The only available quantitative characterization is in terms of programmers average experience in years, given by Takahashi and Kamayachi [22]. Their model can take into account programming experience of up to 7 years, each year reducing the number of defects by about 14%. The data in the study reported by Takada et al [23] suggests that programmers can vary in debugging efficiency by a factor of 3. In a study about the PSP process [19], the defect densities in a program separately written by 104 programmers were evaluated. For about 90% of the programmers, the defect density ranged from about 50 to 250 defects/KSLOC. This suggests that defect densities due to different programming skills can differ by a factor of 5 or even higher.

Thus we propose the model in in Table 3. The skill level may depend on factors other than just the experience. The PSP data suggests while there may be some dependence on experience, programmers with the same experience can have significantly different defect densities.

Team's Average Skill level	Multiplier
High	0.4
Average	1 (default)
Low	2.5

Table 3: The Programming Team Factor (F_{pt})

3.1.3 The Process Maturity Factor (F_m)

This takes into account the rigor of software development process at a specific organization. This level, as measured by the SEI Capability Maturity Model, can be used to quantify it. Here we assume level II as the default level, since a level I organization is not likely to be using software reliability engineering. Kolkhurst [9] assumes that for delivered software, change from level II to level V will reduce defect density by a factor of 500. However, Keene [3] suggests a reduction in the inherent defect density by a factor of 20 for the same change. Jones [7] suggests an improvement by a factor of 4 in potential defects and a factor of 9 in delivered defects for changing from level II to level V. Here we use the numbers suggested by Keene to propose the model given in Table 4.

SEI CMM Level	Multiplier
Level 1	1.5
Level 2	1 (default)
Level 3	0.4
Level 4	0.1
Level 5	0.05

Table 4: The Process Maturity Factor (F_m)

3.1.4 The Software Structure Factor (F_s)

This factor takes into account the dependence of defect density on language type (the fractions of code in assembly and high level languages), program complexity, modularity and the extent of reuse. It can be reasonably assumed that assembly language code is harder to write and thus will have a higher defect density. The influence of program complexity has been extensively debated in the literature [8]. Many complexity measures are strongly correlated to software size. Since we are constructing a model for defect density, software size has already been taken into account. There is some evidence that for the same size, modules with significantly higher complexity are likely to have a higher number of defects. However, further studies are needed to propose a model. It is known that module size influences defect density with a module [2]. However in a software system consisting of modules, the variability due to different block sizes may cancel out if we are considering the average defect density. The influence due to reuse will depend on its extent, the defect-contents of reused modules and how well the reused modules implement the intended functionality. As this time, we propose a model for F_s depending on language use, and allow other factors to be taken in to account by calibrating the model.

$$F_s = 1 + 0.4a \tag{14}$$

where a is the fraction of the code in assembly language. Here we are assuming that assembly code has 40% more defects [1].

3.1.5 Calibrating and using the defect density model

The model given in equation 13 provides an initial estimate. It should be calibrated using past data from the same organization. Calibration requires application of the models using available data in the organization and determining the appropriate values of the subparameters. Since we are using the beginning of the subsystem test phase as the default, Musa et al.'s data suggests that the constant of proportionality C can range from about 6 to 20 defects per KSLOC. For best accuracy, the past data used for calibration should come from projects as similar to the one for which the projection needs to be made. Some of indeterminacy inherent in such models can be taken into account by using a high estimate and a low estimate and using both of them to make projections.

Example 1: For an organization, the value of C has been found to be between 12 to 16. A project is being developed by an average team and the SEI maturiy level is II. About 20% of the code is in assembly language. Other factors are assumed to be *average*.

Then the defect density at the beginning of the subsystem test phase can range between $12 \times 2.5 \times 1 \times 1 \times (1 + 0.4 \times 0.2) = 32.4$ /KSLOC and $16 \times 2.5 \times 1 \times 1 \times (1 + 0.4 \times 0.2) = 43.2$ /KSLOC.

3.1.6 Estimation of β_0^E and β_1^E

Since β_0^E represents the total number of faults that will be detected, it can be estimated using the estimate for the initial defect density, D_0. As suggested by Musa et al., we can assume that about 5% new defects would be created during debugging. Thus we can use this model for β_0^E.

$$\beta_0^E = 1.05 \times D_0 I_s \tag{15}$$

The estimator of β_1^E requires the use of the equation $\beta_1^E = \frac{\hat{K}}{T_L}$ where $\hat{K}$ is the overall value of the fault exposure ratio during the testing period. The value of $\hat{K}$ is some times approximated by 4.2×10^{-7} failures per fault, the average value determined by Musa et al. [18]. Li and Malaiya [11] have suggested that $\hat{K}$ varies with the initial defect density and have given this expression to estimate $\hat{K}$. $\hat{K} = \frac{1.2\times 10^{-6}}{D_0} e^{0.05 D_0}$ where D_0 is the defect density per KSLOC. The parameter values have been computed here by fitting the values for fault exposure ratio for several projects reported by Musa et al. [18].

Example 2: Let us assume that the initial defect density for a project has been estimated to be 25 faults/KSLOC and the software size is 5400 lines. The program is tested on a CPU that runs at 4 MIPS and each source instruction compiles into 4 objects instructions. Then the estimated values are

$$\beta_0^E = 1.05 \times 25 \times 5.4 = 141.7 \tag{16}$$

$$\hat{K} = \frac{1.2 \times 10^{-6}}{25} e^{0.05\times 25} = 1.675 \times 10^{-7} \tag{17}$$

$$\beta_1^E = \frac{1.675 \times 10^{-7}}{\frac{5400\times 4}{4{,}000{,}000}} = 3.10 \times 10^{-5} \tag{18}$$

3.2 Estimation of Logarithmic Model Parameters

Estimating the parameter values for the logarithmic model is a significant challenge. We can take one of two possible approaches. In the first approach we can first estimate the parameters of the exponential model and then compute β_0^L and β_1^L. In the second approach we can calculate β_0^L and β_1^L from the interpretation introduced in section 3.1.

3.2.1 Estimation through β_0^E and β_1^E

Here we use the observation that for a given data set, there is some relationship between β_0^E and β_0^L, and β_1^E and β_1^L. To obtain this relationship, let us assume that both models project the same $\mu(t_f)$ where t_f is the end of the testing period. Let the number of defects remaining at time t_f be $\frac{N_0}{\alpha}, \alpha > 1$. Then

$$\mu(t_f) = N_0 - \frac{N_0}{\alpha} = N_0(1 - \frac{1}{\alpha}) \tag{19}$$

For the exponential model the equation 19 will give,

$$\beta_0^E(1 - e^{-\beta_1^E t_f}) = N_0(1 - \frac{1}{\alpha})$$

since $N_0 \approx \beta_0^E$, we can rewrite this equation as

$$t_f = \frac{ln(\alpha)}{\beta_1^E} \tag{20}$$

using the logarithmic model we can write equation 19 as

$$\beta_0^L ln(1 + \beta_1^L t_f) = N_0(1 - \frac{1}{\alpha})$$

which can be rearranged as

$$t_f = \frac{1}{\beta_1^L}\left[e^{\frac{\beta_0^E}{\beta_0^L}(1-\frac{1}{\alpha})} - 1\right] \tag{21}$$

Equating the right hand side of equations 20 and 21, and rearranging we get

$$\frac{\beta_0^E}{\beta_0^L} = \frac{1}{1 - \frac{1}{\alpha}} ln\left[\frac{\beta_1^L}{\beta_1^E} ln(\alpha) + 1\right] \tag{22}$$

Let us now assume that in time t_f the failure intensity also declines by factor α. Thus according to the exponential model,

$$\beta_0^E \beta_1^E e^{-\beta_1^E t_f} = \frac{\beta_0^E \beta_1^E}{\alpha}$$

which can be solved for to give

$$\beta_1^E = \frac{1}{t_f} ln(\alpha) \tag{23}$$

Similarly the logarithmic model gives

$$\frac{\beta_0^L \beta_1^L}{1+\beta_1^L t_f} = \frac{\beta_0^L \beta_1^L}{\alpha}$$

which can be written as

$$\beta_1^L = \frac{1}{t_f}(\alpha - 1) \tag{24}$$

From equation 23 and 24 we obtain

$$\frac{\beta_1^L}{\beta_1^E} = \frac{\alpha - 1}{ln(\alpha)} \tag{25}$$

Thus is we know α and the values for β_0^E and β_1^E, we can calculate β_1^L using equation 25 and then β_0^L using equation 22.

Example 3: For a software system under test, the parameters β_0^E and β_1^E have been estimated to be 142 and 0.35×10^{-4} respectively. Testing will be continued until about 92% of all faults have been found. That gives

$$\alpha = \frac{100}{100-92} = 12.5 \tag{26}$$

The equation 25 gives

$$\frac{\beta_1^L}{\beta_1^E} = 4.55 \quad i.e. \quad \beta_1^L = 4.55 \times 0.35 \times 10^{-4} = 1.59 \times 10^{-4} \tag{27}$$

and equation 22 gives

$$\frac{\beta_0^E}{\beta_0^L} = 2.75 \quad i.e. \quad \beta_0^L = \frac{142}{2.75} = 51.6 \tag{28}$$

3.3 Direct Estimation of β_0^L and β_1^L

An alternative to the above method is to use the interpretation of β_0^L and β_1^L in terms of D_{min} and K_{min} as given by equation [5]. A reasonable estimate for K_{min} is 1.5×10^{-7} as suggested by the data given by Musa et al. [18]. As estimation of D_{min}, the defect density at which the minimum value of K occurs is harder to estimate. First the curve for K, as shown in figure 1 has a very flat minimum. That can make exact determination of D_{min} hard in the presence of normal statistical fluctuations. Secondly, the variation in K depends on the testing strategy used.

Available data suggest the following.

1. If the initial defect density D_0 is less than 10 per KSLOC, the value of D_{min} is in the neighborhood of 2 defects/KSLOC.

2. However if D_0 is higher, the resulting value of $Dmin$ is also higher. in many cases, taking $D_{min} = D_0/3$ yields a suitable first estimate.

Example 4: For the T2 data [18], the initial defect density is 8.23 defects/KSLOC and the size is approximately 6.92 KSLOC (27.7K object lines). The instruction execution rate is not given in [18], however we can obtain the value of T_L using available information. Since Musa et al. have given the value of $\hat{K}$ as 2.15×10^{-7} and the value of β_1^E can be caluated to be 1.42×10^{-5}, the value of T_L is $2.15 \times 10^{-7}/1.42 \times 10^{-5} = 1.51 \times 10^{-2}$. We will estimate the values of the logarithmic model parameters assuming $D_{min} = 2$ and $K_{min} = 1.5 \times 10^{-7}$.

From equation 11 we have these estimates,

$$\beta_0^L = I_s D_{min} = 6.92 \times 2 = 13.84 \tag{29}$$

and

$$\begin{aligned} \beta_1^L &= \frac{K_{min}}{e} \frac{r}{Q_x I_s} e^{\frac{D_0}{D_{min}}} \\ &= \frac{1.5 \times 10^{-7}}{2.72} \frac{1}{1.5 \times 10^{-2}} e^{\frac{8.23}{2}} \\ &= 2.24 \times 10^{-4} \end{aligned} \tag{30}$$

Fitting of actual test data yields the two values as 17.26 and 2.01×10^{-4}. Considering the fact that the few early points in the test data can often yield values that can be easily off by an order of magnitude or can be illegal (negative), the estimates are quite good.

3.4 Variability of the parameter values

For a give data set, if we use the partial data set from beginning to some intermediate point in testing, the parameter values are found to be different from the final values. We have investigated the incremental variation of the values determined as testing continues. In the beginning the values can change rapidly but later they start settling towards the final value. For practically all data sets, the values of β_0^E and β_0^L rise with testing time whereas for β_1^E and β_1^L the values fall.

The typical behavior is illustrated by the plots for the T1 data. Figure 2 shows that while the value of β_0^E keeps rising, β_0^L appears to stabilize in the later phases of testing. This suggests that the logarithmic model describes the underlying process better. Figure 3 shows how β_1^E and β_1^L vary as testing progreses. Both show a downward trend, however the curve for β_1^L appears to be stabilizing. Figure 4 show the peaks in β_0^L and β_1^L which are largely due to changes in the reliability growth behavior. They are often caused by changes in the testing strategy or by switching to a different test suite. Fortunately often the two parameters are perturbed in the opposite directions, thus minimizing the effect.

The presence of a significant trend in the plots for the exponential model seems to suggest that it does not model the testing process as well as the logarithmic model. All SRGMs are simplified models and hence describe the reliability growth approximately.

The a priori estimates of these models can be better than the values obtained in the early phases of testing, but can not be expected to be as accurate as the final values obtained using actual test data.

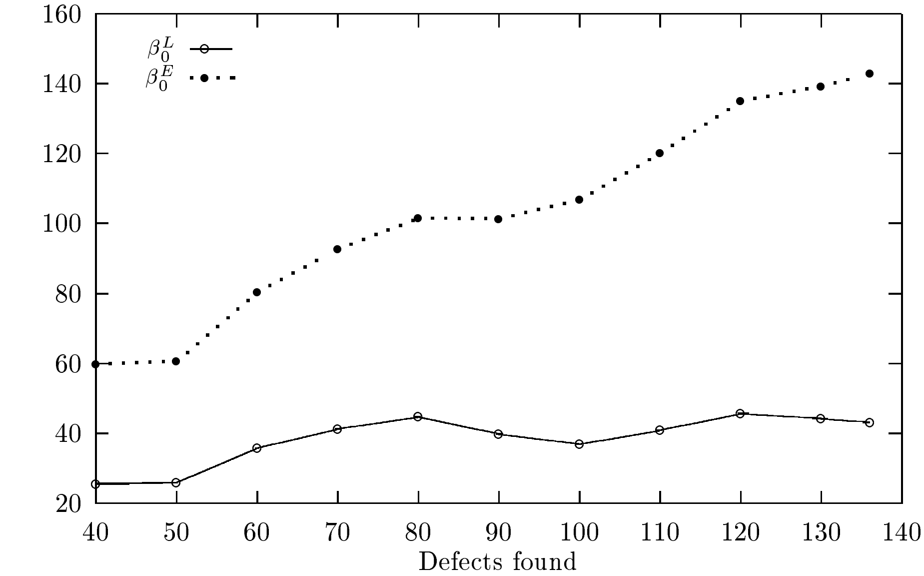

Figure 2: Variation of β_0^E and β_0^L

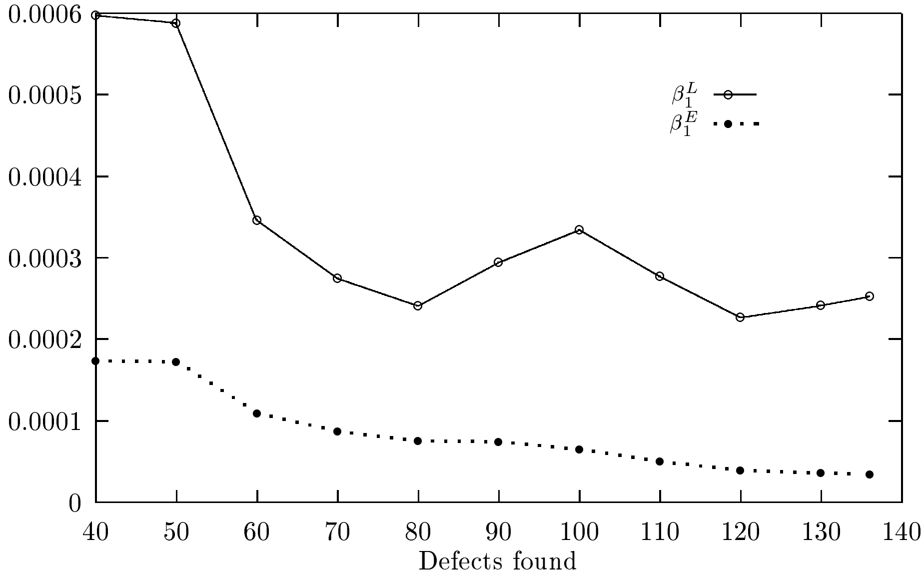

Figure 3: Variation of β_1^E and β_1^L

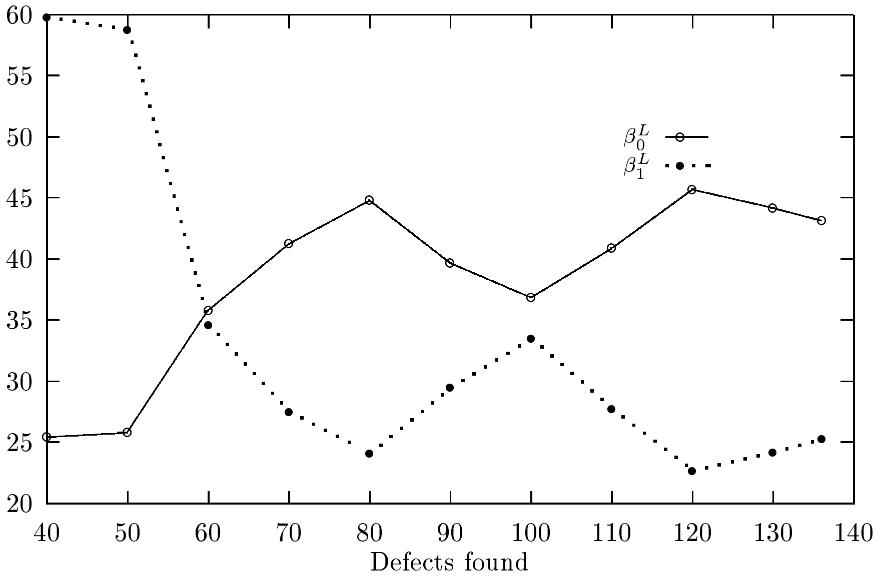

Figure 4: Variation of β_0^L and β_1^L (*rescaled*)

4 Concluding Remarks

In this study we have presented methods to estimate the parameters of the exponential and the logarithmic models. We have proposed an empirical model for estimating the defect density. A new interpretation for the parameters of the logarithmic model has been proposed and we have shown how it can be used to estimate the values. An alternative approach is to first estimate the parameters for the exponential model and them use them to estimate the logarithmic model parameters.

The methods presented here can significantly improve the accuracy of the projections during the early phases of testing. The accuracy of the results will depend on careful calibration of the models using data from projects that have used a similar process.

Future work includes a detailed analysis of the specific results for the data sets available. We also need to investigate the sensitivity of the projections due to variation in the parameters values.

References

[1] J.R. Adam, “Software Reliability Predictions are Practical Today”, Proc. IEEE Ann. Symp. on Software Reliability, Colorado Springs, May 1989.

[2] W. W. Agresti and W. M. Evanco, “Projecting Software Defects from Analyzing Ada Designs,” IEEE Trans. Software Engineering, Nov. 1992, pp. 288-297.

[3] G.F. Cole and S.N. Keene, “Reliability and Growth of Fielded Software,” Reliability Review, March 1994, pp. 5-26.

[4] W. Farr, Software Reliability Modeling Survey, in *Handbook of Software Reliability Engineering*, Ed. M. R. Lyu, McGraw-Hill, 1996, pp. 71-117.

[5] J. Gaffney and J. Pietrolewicz, "An Automated Model for Early Error Prediction in Software Development Process," Proc. IEEE Software Reliability Symposium Colorado Spring, June 1990.

[6] M. Gechman and K. Kao, "Tracking Software Reliability and Reliability with Metrics," Proc. ISSRE Industry Reports, 1994.

[7] C. Jones, "Software Benchmarking" Web Document, IEEE Computer, Oct. 1995. http://www.computer.org/pubs/computer/software/10/software.htm.

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

[9] B.A. Kolkhurst, "Perspectives on Software Reliability Engineering Approaches found in Industry" Proc. ISSRE Industry Reports, 1994.

[10] N. Li and Y.K. Malaiya "ROBUST: A Next Generation Software Reliability Engineering Tool" Proc. IEEE Int. Symp. on Software Reliability Engineering, pp. 375-380, Oct. 1995.

[11] N. Li and Y.K. Malaiya, ROBUST: A Next Generation Software Reliability Engineering Tool, Proc. IEEE Int. Symp. on Software Reliability Engineering, pp. 375-380, Oct. 1995.

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

[13] N. Li, "Measurement and Enhancement of Software Reliability Through Testing," Ph.D. dissertation, Colorado State University, 1997.

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

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

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

[17] J. D. Musa and K. Okumoto, A Logarithmic Poisson Execution Time Model for Software Reliability Measurement, *Proc. 7th Int. Conf. on Software Engineering*, 1984, pp. 230-238.

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

[19] "Personal Software Process" Web Document, Carnegie Mellon University, http://www.sei.cmu.edu/technology/psp/Results.htm, Rev. 5 Sept. 1997

[20] P. Piwowarski. M. Ohba and J. Caruso, "Coverage measurement Experience during Function Test," Proc. ICSE, 1993, pp. 287-301.

[21] Rome Lab, "Methodology for Software Reliability Prediction and Assessment," Tech Report RL-TR-95-52, Vol. 1 and 2, 1992.

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

[23] Y. Tokada, K. Matsumoto and K. Torii, "A programmer Performance Measure based on Programmer State Transitions in Testing and Debugging Process," Proc. International Conference of Software Engineering, 1994, pp. 123-132.