Predictability of Software Reliability Models*

Yashwant K. Malaiya, Nachimuthu Karunanithi

Colorado State University, Fort Collins, CO 80523.

Pradeep Verma

Hewlett-Packard, Information Network Division, CA 95014.

Key Words – Software Reliability Growth Models, Predictability, Model Comparison

Readers Aid –

Purpose: Widen State Of Art

Special math needed for explanation: Probability, Simple Calculus

Special math needed to use results: Same

Results useful to: Software reliability theoreticians, Software Managers

Abstract – For software projects it is critical to be able to achieve a minimum quality level before they are released. It is often important to meet a target release date. To be able to estimate the testing efforts required, it is necessary to use a *software reliability growth model*. While several different software reliability growth models have been proposed, few guidelines exist about which model should be used. Here a two-component predictability measure is presented that characterizes the long term predictability of a model. The first component, *average predictability*, measures how well a model predicts *throughout the testing phase*. The second component, *average bias*, is a measure of the general tendency to overestimate or underestimate the number of faults. Data sets for both large and small projects from diverse sources have been analyzed. Results presented here indicate that some models tend to perform better than others.

*This work was supported from a project funded by SDIO/IST and monitored by ONR

1 INTRODUCTION

Establishing the quality of software systems has become one of the major challenges in all software production environments. A software product can be released only after some threshold reliability criterion has been satisfied. It is necessary to use some heuristics to estimate the required test time so that available resources can be efficiently apportioned. The most useful reliability criteria are residual fault density or the failure intensity. One of the best approaches to determine the required testing time is to use a time based Software Reliability Growth Model (SRGM). In recent years researchers have proposed several different SRGMs. A comprehensive survey and classification of software reliability models can be found in [?, ?, ?].

All software reliability models are based on some key assumptions about the environment and they model different failure processes. There is evidence to suggest that they have different prediction capabilities, specially during early phases of testing. This is the duration when better predictability is required to estimate the release date and the additional test effort required. Hence selection of a particular model can be important for a reliable estimate of reliability of software systems.

Notation

$\lambda(t)$ failure intensity (i.e, fault detection rate), at time t, derivative of $\mu(t)$

$\mu(t)$ is the *expected* number of failures experienced in $(0, t)$

λ_i observed value of λ at instant t_i

μ_i observed value of μ at instant t_i

n number of observation points

β_0, β_1 parameters characterizing a fault model

M_k a model $\in$ { LOGM, POWM, INPM, EXPM, DSSM}

D_j actual total number of faults detected in data set j

D_{ij}^k predicted total number of faults to be detected by using

model k, and part of the data set j corresponding to $t_0, ..., t_i$.

T_i time between (i-1) and ith failures.

2 SOFTWARE RELIABILITY GROWTH MODELS

Here five different fault count models are considered. The most common approach is to use a grouped data. The testing duration is divided into a number of periods. For each period, one item of the data set (t_i, λ_i), or equivalently (t_i, μ_i) is obtained. The major objective of using a model is to be able to estimate the time t_F when the failure intensity $\lambda(t_F)$ would have fallen below an acceptable threshold.

Five of the most commonly used execution time SRGMs have been examined here. The *exponential* model [**?**, **?**] with its variations is one of the most commonly used models. The *logarithmic* model proposed by Musa and Okumoto [**?**] is one of the more recent models. The *Delayed S-shaped* model proposed by Yamada et al.[**?**] is one of the recent addition to the family of Gamma distribution models. In addition, we have also examined the *inverse-polynomial* model proposed by Littlewood and Verrall [**?**] and the *power* model by Crow [**?**]. All these models are two parameter models. This allows a fair comparison among the models. It was felt that these models do represent a sufficiently wide range of presumed behavior. All the models considered are NHPP (Non-Homogeneous Poisson Process) models with the exception of *inverse-polynomial* model.

The fault models are described below. For some models, the parameters have a specific interpretation. We first describe the important steps involved in parameter estimation of the *logarithmic* model. Since the same steps can be applied for other models we only show their basic equations. Since the number of data points is not large we have used the *least squares* technique in our experiments. The *maximum likelihood* method has been found to perform similarly in this application [**?**].

Logarithmic Model (LOGM): This model was proposed by Musa and Okumoto [**?**]. Here the underlying software failure process has the characteristics of a logarithmic poisson process and it assumes the total number of failures in the system to be equal to infinity in infinite time. It has an intensity function that decreases exponentially with the number of

failures experienced. The mean value function and the failure intensity are [?]:

$\mu(t;\beta) = \beta_0 \ln(1 + \beta_1 t)$

$\lambda(t;\beta) = \frac{\beta_0 \beta_1}{1+\beta_1 t}$

It should be noted that the failure intensity can also be expressed as:

$\lambda(\mu;\beta) = \beta_0 \beta_1 exp(-\frac{\mu}{\beta_0})$.

The square of the sum of the errors, S is given by:

$S(\beta_0, \beta_1) = \sum_{l=1}^{n} [\ln r_l - \ln \beta_0 \beta_1 + \ln(1 + \beta_1 t)]^2$

where r_l is the actual failure intensity at t_l, calculated from the input data. Minimizing this expression results in the least square estimation of the parameters β_0 and β_1.

It is easily seen that $\beta_0 \beta_1$ represent the initial failure intensity at time 0. This model belongs to the infinite-fault category [?], and thus the concept of an initial number of faults does not exist.

Inverse Polynomial Model (INPM): This model was proposed by Littlewood and Verrall [?]. This model is more general and flexible enough where one can choose different reliability growth functions. In our experiment we have considered the model with a second degree polynomial. The main characteristic feature of this model is that the program hazard rate decreases with time and experiences discontinuities of varying heights at each failure. The mean value function and failure intensity equations are [?]:

$\mu(t;\beta) = 3\beta_0 (Q1 + Q2)$

$\lambda(t;\beta) = \frac{\beta_0}{\sqrt{t^2+\beta_1}} (Q1 - Q2)$

where,

$Q1 = \sqrt[3]{t + (t^2 + \beta_1)^{1/2}}$ and $Q2 = \sqrt[3]{t - (t^2 + \beta_1)^{1/2}}$

Although this is not a popular model, it was chosen for examination, because in [?], it has been shown to have good predictability for one data set.

Exponential Model (EXPM): This model was originally proposed by Moranda [?]

and Musa [?] reformulated it in terms of execution time. Several models can be shown to be variations of this model[?]. Here the important assumption is that the per fault hazard rate is a constant. The mean value function and failure intensity equations are:

$$\mu(t;\beta) = \beta_0[1 - exp(-\beta_1 t)]$$

$$\lambda(t;\beta) = \beta_0\beta_1 exp(-\beta_1 t)$$

This is a finite-failures model. The parameter β_0 represents the initial number of faults and $\beta_0\beta_1$ is the initial failure intensity. Techniques exists to estimate β_0 and β_1 empirically [?].

Power Model (**POWM**): This model was proposed by Crow [?] to estimate reliability of hardware systems during development testing. This model models the failure events as a nonhomogeneous Poisson process whose failure intensity function is a power function of time. However, it is possible to apply this model to estimate software reliability by controlling the failure intensity range. For the purpose of our experiment we have kept the value of $\beta_1 < 1.0$. The basic equations of the mean value function and the failure intensity are:

$$\mu(t;\beta) = \beta_0 t^{\beta_1}$$

$$\lambda(t;\beta) = \beta_0\beta_1 t^{\beta_1 - 1}$$

This is a infinite-failures model.

Delayed S-Shaped Model (**DSSM**): This model was proposed by Yamada, Ohba, and Osaki [?]. This model characterizes software failure process as a delayed S-shaped growth. Here software error detection process can be regarded as a learning process in which test-team members improve their familiarity and skill gradually. This is regarded to be the best of the several S-Shaped models[?]. The basic equations for the mean number of failure value and the failure intensity are:

$$\mu(t;\beta) = \beta_0[1 - (1 + \beta_1 t)e^{-\beta_1 t}]$$

$$\lambda(t;\beta) = \beta_0\beta_1^2 te^{-\beta_1 t}$$

This is a finite-failures model. The total number of faults is β_0 and $1/\beta_1$ is the time when the maximum failure intensity occurs. Most other models assume that the failure intensity

starts declining from the beginning. However the S-Shaped models can represent an initial rise in failure intensity that is sometimes observed.

Figures 1 and 2 show the typical behavior of these models in terms of μ and λ.

.

3 A NEW PREDICTABILITY MEASURE

Even though a large number of models are available in the literature, no clear guidelines have been presented for selection of a particular model. Characterization of a model requires its application to a number of data sets and evaluating its predictive capabilities. Applicability of a model can be measured using one of the three distinct approaches considered below.

1.**Goodness-of-fit**: This may be termed as an end-point approach because evaluation can be done only after all the data points $\{t_i, \lambda_i\},\ i = 0, 1, ..., n$, or equivalently $\{t_i, \mu_i\},\ i = 0, 1, ..., n$ are available. After a curve corresponding to a selected model M_k is fitted to the data, the deviation between observed and the fitted values is evaluated by using the Chi-

Square test or Kolmogorov-Smirnov[?] test. The Kolmogorov-Smirnov test is considered to be more effective compared with the Chi-Square test. The goodness-of-fit approach has low computational requirements and is the one used most widely. For example, it has been used by Matsumoto et al. to compare the Exponential, Hyperexponential and the S-Shaped models [?].

The disadvantage with goodness-of-fit measures is that they do not measure predictability. It is possible to have a model which fits the later behavior but not earlier. Such a model can have a good overall fit while providing poor predictability in the early phases.

2.**Next-Step-Predictability**: In this approach a partial data set, say, $\{t_i\},\ i = 1, ..., (l-1)$, is used to predict the value of T_l. The predicted and the observed values of T_l are then compared. This approach has been used by Abdel-Ghaly et al.[?] and Brocklehurst et al.[?]. This method can also be used for grouped data. This approach allows predictability to be measured with a partial data set, while testing is still in progress. However it only measures short-term predictability.

3.**Variable-Term-Predictability**: Short-term predictability can be an appropriate measure near the end of the test phase. However in actual practice, the need to predict the behavior near the end of the test phase by using data available near the beginning of the test phase, is very important. This approach, used by Musa, Iannino and Okumoto [?] and Malaiya et al.[?], makes projections of the final value of μ at each t_l, using the partial data set $\{t_i, \mu_i\},\ i = 0, ..., l$. The error in these projections can then be plotted against time. This method requires n different curve-fitting for each data set. The effectiveness of the weighted-parameter estimation has been examined by Verma and Malaiya[?] using the same approach. Sukert [?] has empirically validated Jelinski-Moranda, Schick-Wolverton, and modified Schick-Wolverton models using data sets from four DOD projects. However he has not presented any formal comparison.

In this paper the variable-term predictability approach has been used to compare some of the major SRGMs. Actual data from a wide spectrum of software projects, corresponding to different initial fault densities have been used.

Our proposed *two-component predictability measure* consists of *Average Error* (AE) and *Average Bias* (AB). The AE is a measure of how well a model predicts throughout the test phase. The AB indicates the general bias of the model. The AB can be either positive or negative depending on whether the model is prone to overestimation or underestimation.

More formally, let the data be grouped into n points $(t_i, \lambda_i),\ i = 1\ to\ n$, where λ_i is the failure intensity at time t_i. Using a specific model M_k, and data set j, let D_{ij}^k be the projected total number of the faults to be detected. Thus

$$D_{ij}^k = \mathcal{F}\{M_k; (t_1, \lambda_1), (t_2, \lambda_2), \cdots, (t_i, \lambda_i)\}.$$

Taking the variable-term approach[**?**], we can make plots of prediction error $(D_{ij}^k - D_j)/D_j,\ i = 1$ to n, for each model k. Then predictability measures are given by

$$(i)\ AE_j^k = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{D_{ij}^k - D_j}{D_j}\right|$$

$$(ii)\ AB_j^k = \frac{1}{n}\sum_{i=1}^{n}\frac{D_{ij}^k - D_j}{D_j}$$

The use of these measures is illustrated in Figure 3. The total shaded area under the curve is used for computing AE_j^k. On the other hand, calculating AB_j^k requires use of the appropriate sign.

The different data sets used correspond to various fault density ranges. To have a proper comparison, we have trimmed some of the data so that they correspond to one of a few selected ranges. Let t_l and t_u be the time corresponding to lower and upper bound fault densities. Our lower limits correspond to a point where the relative error is significantly high. Our upper limits correspond to a point where sufficient faults have been detected (and corrected) and the system has achieved a high reliability. In our comparison scheme we calculate ABs and AEs only within these limits. These limits are specific to each data set and do not depend on the model used. Let m_j be the total number of points between t_l and t_u for each data set j. Hence,

$(i)\ AE_j^k = \frac{1}{m_j} \sum_{i=l}^{u} \left| \frac{D_{ij} - D_j}{D_j} \right|$

$(ii)\ AB_j^k = \frac{1}{m_j} \sum_{i=l}^{u} \frac{D_{ij} - D_j}{D_j}$

Our observations show that the omission of end-points do not affect the generality of the conclusions drawn here.

For the purpose of comparison, the data sets examined (Table 1) can be classified into one of the following ranges: 1) *High Fault Density Range* in which the initial fault density is more than 10.0 faults/KLOC, 2) *Medium Fault Density Range* in which the initial fault density is between 1.0 and 10.0 faults/KLOC, and 3) *Low Fault Density Range* in which the initial fault density of the systems is less than 1.0 faults/KLOC. The first range is commonly encountered at the beginning of unit test or the system test phase[**?**]. The middle range corresponds to systems that are either in system test phase or in some cases, in the operational phase. The last range corresponds to software which has very low fault density to start with and is normally encountered in operational mode. Comparison of these models in these ranges would allow us to see if the predictability of a model has any significant dependence on fault density. Since the data sets are for diverse projects from diverse teams, observations are more useful. If the data sets are from a single environment, the general

applicability of the results may be less reliable.

TABLE 1. THE DATA SETS USED.

Data Sets	Reference	Code Size Lines	Faults Detected	Software Type	Range Used faults/KLOC
1.1	[?]	1000	27	Class Compiler Project	20.0–5.0(H)
1.2	”	1000	24	”	”
1.3	”	1000	21	”	”
1.4	”	1000	27	”	”
2	[?]	21,700	136	Realtime Command and Control	”
3.1	[?]	40,000	46	On-line Data Entry	1.00–0.05(M)
3.2	[?]	1,317,000	328	Database Application Software	0.20–0.05(L)
3.3	[?]	35,000	279	Hardware Control Software	7.0–0.7(M)
4	[?]	180,000	101	Military Application Software	0.20–0.05(L)
5	[?]	240,000	3207	Application Software	13.0–1.3(H)
6	[?]	870,000	535	Realtime Control Application	0.5–0.05(L)
7	[?]	200,000	481	Monitoring and Realtime Control	2.0–0.05(M)
8	[?]	14,500	55	Railway Interlocking System	3.0–0.05(M)
9	[?]	90,000	198	Monitoring and Realtime Control	2.0–0.05(M)
10.1	[?]	10,000	118	Flight Dynamic Application	10.0–1.0(H)
10.2	[?]	22,500	180	Flight Dynamic Application	7.0–0.7(M)
10.3	[?]	38,500	213	Flight Dynamic Application	5.0–0.5(M)
11	[?]	1,000,000	231	Not Known	0.20–0.05(L)

TABLE 2. AVERAGE ERROR OF MODELS(AEs) AT DIFFERENT FAULT DENSITY RANGES.

Data Set Used	MODELS					Data Set Average
	Log	Inv Poly	Exponen	Power	S-Shaped	
HIGH FAULT DENSITY RANGE						
1.1	23.69	28.04	36.32	16.31	44.82	29.84
1.2	36.00	17.31	44.00	65.27	30.83	38.68
1.3	22.22	33.08	31.36	12.26	47.57	29.30
1.4	18.96	36.87	24.24	41.87	47.60	33.91
2	10.58	8.11	37.48	34.26	47.60	27.41
5	8.03	8.54	9.06	13.04	22.73	12.28
10.1	11.48	12.14	17.58	23.68	16.59	16.29
MEDIUM FAULT DENSITY RANGE						
3.1	21.58	24.77	25.19	24.32	30.48	25.27
3.3	8.80	37.64	13.33	45.50	30.92	27.24
7	13.58	25.69	30.87	28.97	14.38	22.70
8	4.95	9.40	20.81	37.18	22.28	18.92
9	14.64	31.50	14.98	49.97	33.18	28.85
10.2	10.09	24.76	17.84	18.66	25.26	19.32
10.3	13.39	19.64	29.12	19.40	35.21	23.35
LOW FAULT DENSITY RANGE						
3.2	22.35	32.36	30.49	36.83	31.69	30.74
4	33.11	47.63	15.66	66.42	31.29	38.82
6	17.43	19.10	21.47	34.91	28.00	24.18
11	26.29	24.23	33.50	15.29	43.16	28.49

TABLE 3. AVERAGE BIAS OF MODELS (ABs) AT DIFFERENT FAULT DENSITY RANGES.

Data Set Used	MODELS					Data Set Average
	Log	Inv Poly	Exponen	Power	S-Shaped	
HIGH FAULT DENSITY RANGE						
1.1	-23.69	-20.41	-36.32	-2.41	-44.82	-25.53
1.2	-36.00	+17.31	-44.00	+65.27	-11.41	-1.77
1.3	-22.22	+3.11	-31.36	-4.55	-26.32	-16.27
1.4	-9.14	+26.01	-18.21	+37.44	-29.52	+1.32
2	-10.04	+5.18	-37.48	+33.57	-47.60	-11.27
5	+6.34	+4.44	+5.42	+12.57	-21.95	+1.36
10.1	-2.32	-5.72	-13.87	+23.53	-14.76	-2.63
MEDIUM FAULT DENSITY RANGE						
3.1	-21.58	-23.01	-25.19	-9.54	-30.30	-21.92
3.3	-4.31	+32.68	-12.89	+45.49	-17.39	+8.72
7	+11.23	+24.02	+20.51	+28.40	-10.95	+14.64
8	-3.49	+0.97	-13.88	+37.16	-10.06	+2.14
9	+14.64	+31.50	+12.21	+49.97	-33.18	+15.03
10.2	-1.67	+7.59	-12.07	+10.16	-20.41	-3.28
10.3	-13.39	-19.64	-29.12	+8.62	-35.21	-17.75
LOW FAULT DENSITY RANGE						
3.2	-18.00	+6.07	-24.94	+18.71	-28.70	-9.37
4	+25.86	+44.54	-14.77	+61.15	-31.29	+17.10
6	+14.68	+9.33	+3.86	+34.91	-27.06	+7.14
11	-24.89	-19.57	-32.12	-12.21	-43.16	-26.39

TABLE 4a. WEIGHTED AVERAGE ERROR OF MODELS AT DIFFERENT FAULT DENSITY RANGES.

Fault Density Range	MODELS				
	Log	Inv Poly	Exponen	Power	S-Shaped
HIGH	8.6	8.9	12.0	15.4	24.8
MEDIUM	13.7	26.6	24.6	33.1	23.6
LOW	22.8	27.3	28.3	31.5	34.1
Overall Average	20.9	26.0	26.8	30.6	32.4

TABLE 4b. WEIGHTED AVERAGE BIAS OF MODELS AT DIFFERENT FAULT DENSITY RANGES.

Fault Density Range	MODELS				
	Log	Inv Poly	Exponen	Power	S-Shaped
HIGH	+4.3	+4.2	+0.8	+14.8	-23.8
MEDIUM	+4.4	+16.6	+4.9	+28.4	-20.3
LOW	-9.3	+1.4	-19.1	+16.0	-32.7
Overall Average	-6.9	+3.2	-15.2	+17.2	-30.8

4 COMPARISON OF MODELS

In our analysis we have used 18 different data sets collected from a wide variety of software systems [?, ?, ?, ?, ?, ?, ?, ?, ?, ?] as shown in Table 1. They range from a compiler project in an academic environment to a set of hardware control modules used in a practical system. Among them 8 data sets are from Japanese software projects. Note that data sets 3.1, 3.2, 5, 6, 7, 8 and 9 are calendar time based while the remaining data sets are execution time based. The size of the source code for the data sets 2, 5 and 8 were reported in assembly instructions whereas for the remaining data sets it was in high level languages. In the last column of the Table 1 we use the following notations to represent different fault density

ranges: H for high, M for medium and L for low. While in this study we have used more data sets than most other studies, there is a need to collect even more extensive number of data sets from diverse sources and repeat the examination. Performances of these models, in terms of AE and AB, are shown in Tables 2 and 3, and in Figures 4 and 5. In Table 2, a smaller value of AE implies a higher accuracy in predictability of a given model. The AB values in Table 3 indicate whether a model is prone to overprediction or underprediction.

4.1 COMPARISON WITH WEIGHTS

In this section it is assumed that the results from the large projects have more significance, hence we have included a weighting factor corresponding to the actual size of the software when averaging. Table 4a represents the weighted average and overall predictive capabilities of these models. Table 4b summarizes the weighted bias measure of these models. Figure 4 highlights the average error in predictability of models at different fault density ranges across all data sets. Figure 5 is used to indicate how these models exhibit bias at various fault density ranges.

.

The *logarithmic model*, as shown in Tables 2, seems to perform relatively well. Though it is not the best predictor in all cases, it has projected the remaining faults most accurately in the medium fault density range. Also in the high and the low fault density ranges its AE measures are better than other models in majority of the cases. When the best values are predicted by other models in these two ranges, the LOGM's AEs are often close. As shown in Tables 3 its AB values indicate that this model has a mixed bias, sometimes predicting more and sometimes predicting fewer than actual number of faults. The weighted average AE measures given in Table 4a and Figure 4 suggest that the LOGM has performed relatively better than other models. Table 4b indicates that the LOGM exhibited a slight positive

bias in high and medium fault density ranges, although the averaged AB measure over all fault density ranges showed a slight negative bias. Figure 5 summarizes the behavior of the LOGM in terms of average bias. This suggests that the LOGM may offer good predictability during different phases, be it testing or field operations.

The *inverse polynomial model* seems to perform decent predictions. The INPM has the lowest AE value of 17.31 and 7.11 for the data sets 1.2 and 2 in the high fault density range, and its AEs are in the middle range in most of the remaining cases. From Figure 4a it appears that the INPM has the second best predictive capability in the high and the low fault density ranges. From the overall average AE value in Figure 4, the INPM appears to be the second best predictor. Results in Table 3 show that the INPM has the tendency of positive bias in majority of the cases. The values in Table 4b and Figure 5 indicate that this model is a consistent overpredictor in all fault density ranges. However it should be noted that its overall bias has the lowest positive value among all models. This may make it suitable for adaptive implementation or for getting a high estimate.

The *exponential model's* biased predictive behavior is somewhat similar to that of the LOGM. Except for data set 4, where EXPM has the lowest value of 15.66, its AE values in most of the remaining cases are about in the middle. The AEs in Table 4a and Figure 4 indicate that the EXPM is the third best predictor in all fault density ranges. Except for data sets 5, 6, 7 and 9 where it has positive AE values, its negative AB values lie between the extreme values predicted by other models. In many of the cases, the EXPM's relatively higher AB values imply that the EXPM may underproject more compared to the LOGM. From Table 4b we can see that the EXPM shows a small positive bias in the high and the medium fault density ranges and a significant negative bias in the low fault density range. As shown in Figure 5 the overall bias is negative.

The *power model*, as shown in Table 2, seems to perform highly inconsistently. Though the POWM's AE measure of 16.31, 12.26, and 15.29 are the best for data sets 1.1, 1.3, and 11, it has the highest AEs in eight cases (data set: 1.2, 3.2, 3.3, 4, 6, 8, 9 and 10.1). As shown in Table 4a and Figure 4 its average AE is the highest among all models in the medium fault density range and the second highest value in the low fault density range. Even in the

high fault density range its prediction errors are close to the highest values given by the DSSM. This suggests that this model may not be a good predictor once the product reaches higher reliability. A comparison in Table 3 shows that both the INPM and the POWM seem to have similar bias in all cases except for data sets 1.3, 10.1, and 10.3. The POWM's AB values are the highest positive values when compared to that of other models as shown in Table 4a and Figure 5.

The *delayed S-Shaped model*, as shown in Tables 2, seems to be a poor estimator in many cases. Its AE values are the highest for the data sets 1.1, 1.3, 1.4, 3.1, 5, 10.1, 10.3, 2, and 11. The delayed S-Shaped model performs poorly in the high fault density range as shown in Table 4a. In the medium fault density range its performance is not significantly better than that of the worst predictor, the POWM. Overall weighted average AE measure in Figure 4 implies that the DSSM predicts with the highest error among all models. AB values in Table 3 indicate that the DSSM has consistent negative bias across all data sets. Moreover its AB values are the highest negative bias for all data sets except for cases 1.2, 1.3 and 8. Its weighted average AB values in Table 4b and Figure 5 show that the DSSM as the only model that consistently underestimates in all fault density ranges. The DSSM model is stable in early phases of testing which accounts for why some prefer to use it. Overall it might be a weak model. Zinnel [**?**] has shown that a corrective strategy can improve the performance of this model. This needs to be further investigated.

Our observation from Table 3 suggests that performance of these models may sometimes be affected by the peculiarities of the data set. For example, data sets 1.1 and 11 forced all models to exhibit negative bias. One possible approach to overcome this peculiarity can be to use either an adaptive prediction approach [**?**] or a non-parametric approach suggested in [**?**, **?**]. One more observation from Table 4a suggests that all these models, with the exception of the DSSM, have decreasing accuracy in prediction when the fault density is decreased. Further evaluation is needed to verify that this is indeed significant. A possible explanation can be that at low failure intensity, the information content in the available data may be lower.

4.2 COMPARISON WITHOUT WEIGHTS

In the previous section we compared the predictive accuracy of models at different fault density ranges using weighted average AE measures. In order to make our observations more stronger we performed further statistical analysis without assigning weights to the projects. We performed statistical analysis using the ANOVA method. In this approach we viewed the results as outcomes of randomized block experiment in which the projects were randomly selected and the SRGMs as "treatments" applied to each of the projects.

In order to make comparison among the SRGMs we performed an "intra-project" comparisons in the following way. We classified the AE measures into a projects-by-SRGMs two-way table and performed ANOVA with projects being the blocks and the SRGMs the treatments. Note that this approach takes into account the peculiarities in the different projects which could potentially impact the the performance of these competing models.

Since we are also interested in comparing SRGMs at different fault density ranges we performed a variant of the two-way classification ANOVA. In this analysis, we treated observations as outcomes of an unbalanced nested experiment. The results of this ANOVA is shown in Table 5.

TABLE 5. ANOVA RESULTS WITH DIFFERENT FAULT DENSITY RANGES.

Source	df	MSS	$F_{Statistics}$
FD	2	307.0	
Projects within FD	15	242.2	
SRGMs	4	615.6	5.72
FD * SRGMs	8	94.4	0.87
SRGMs * Projects within FD	60	107.6	
(Error)			

To find whether there is any significant difference among the SRGMs, we used the F statistic $\frac{ModelsMSS}{ErrorMSS}$ which takes the value 5.72 with 4 and 60 df and is significant. To check

whether there are significant interactions between the fault density levels and the SRGMs, we used the F statistic $\frac{FD*ModelsMSS}{ErrorMSS}$ which takes the value 0.87 with 8 and 60 df and is significant. Since the interaction is not significant the performance of a model relative to the other models is not affected by fault density. The resulting $\overline{AE}$ are shown in Table 6 and Figure 6.

TABLE 6. AVERAGE ERROR OF MODELS AT DIFFERENT FAULT DENSITY RANGES.

Fault Density Range	MODELS				
	Log	Inv Poly	Exponen	Power	S-Shaped
HIGH	18.7	20.6	28.6	29.5	36.8
MEDIUM	12.4	24.8	21.7	32.0	27.4
LOW	24.8	30.8	25.3	38.4	33.5
Overall Average	17.6	24.5	25.2	32.5	32.4

figure=./fig6.ps,height=3.5in

The average AE values for the models are: Log = 17.6, Inv Poly = 24.5, Exp = 25.1, Power = 32.4 and S-Shaped = 32.4. Since the SRGMs have significant F statistics, the Least Significant Difference (LSD) procedure can be used to differentiate among them. Using the LSD method, two SRGMs are significantly different if their means have a difference of 6.9 or above. The LSD value of 6.9 is calculated using $T_{0.05,68}$ critical equal to 2.0. If the difference is less than 6.9 then we can interpret that the SRGMs have similar predictive accuracy across all projects. Thus we can conclude, using > to denote significantly better, that { Log } > { Inv Poly, Exp } > { Power, S-Shaped }. This conclusion agrees with our earlier observations based on the weighted analysis.

From these graphs we can make the following observations: 1) the performance of these models may vary at different fault density ranges and 2) the Log model may perform relatively well across different fault density ranges. However these observations should be verified further using more data sets and analysis.

5 CONCLUDING REMARKS

We have presented an approach for evaluating predictability of software reliability growth models which is related with how the models are actually used. We have evaluated *average error* and *average bias* for different models for data sets for diverse projects with varying initial fault density ranges. Our results seem to support Musa's observation[?] that the logarithmic model appears to have good predictability in most cases. Furthermore, our overall result suggests that the Inverse Polynomial model can be used as the next alternative. The delayed S-Shaped model, which in some cases have been shown to have good fit, generally performed poorly. The statistical analysis also shows that these models have significantly different predictive capabilities.

However, it should be noted that what we have evaluated is not just the models but rather the combination of our evaluation scheme as well as models. Thus, whenever our approach is used for evaluating a (new) model the effectiveness of our scheme should also be taken into consideration.

Both calendar time and execution time data sets were used in our study. Although one might expect that the S-shape model would have better predictability for calendar time based data sets, we did not notice any significant difference in performance. Furthermore, our results did not indicate any significant difference in the performances of other models due to the use of either calendar time or execution time. This aspect of the problem needs further study.

It is found that some models tend to overestimate or underestimate. It seems that the inverse polynomial model and the power model are prone to consistent overprediction. Our result also suggests that the delayed S-shaped model is a consistent underpredictor. It has been argued that a consistent positive bias should be considered as an optimistic prediction [?]. In addition, some data sets may have peculiarities that may cause most models to have either positive or negative bias. It may be possible to exploit this fact to do an a priori or adaptive correction as shown in [?]. An alternative approach to avoid such biased predictions may be to combine two or more models with opposite biases.

ACKNOWLEDGEMENT: We would like to thank the anonymous reviewer who provided many useful comments and suggested the use of the ANOVA method.

References

[1] A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, "Evaluation of Competing Software Reliability Predictions", *IEEE Trans. Reliability.*, vol SE-12, 1986 Sep, pp 950-967.

[2] B. M. Anna-Mary, " A Study of the Musa Reliability Model", *M. S Thesis, CS Department, University of Maryland.*, 1980.

[3] S. Brocklehurst, P. Y. Chan, B. Littlewood and J. Snell, "Recalibrating Software Reliability Models", *IEEE Trans. Software Eng.*, vol 16, 1990 Apr, pp 458-470.

[4] W. J. Conover, **Practical Non-Parametric Statistics**, John Wiley and Sons Inc., 1971, Chapter 6.

[5] L. H. Crow, "Reliability for Complex Repairable Systems", *Reliability and Biometry, SIAM*, Philadelphia, 1974, pp 379-410.

[6] A. L. Goel, "Software Reliability Models: Assumptions, Limitations and Applicability", *IEEE Trans. Software Eng.*, vol SE-11, 1985 Dec, pp 1411-1423.

[7] N. Karunanithi, Y. K. Malaiya and D. Whitley "Prediction of Software Reliability Using Neural Networks", In the *Proc. of the IEEE Int. Symp. Software Reliability Eng.*, 1991 May, pp 124-130.

[8] N. Karunanithi, D. Whitley and Y. K. Malaiya "Prediction of Software Reliability Using Connectionist Approach", *Tech Report No CS-91-116, Computer Science Dept, Colorado State University*, 1991 Aug.

[9] B. Littlewood and J. L. Verrall, "A Bayesian Reliability Model with a Stochastically Monotone Failure Rate", *IEEE Trans. Reliability.*, vol R-23, 1974, pp 108-114.

[10] Y. K. Malaiya, S. Sur, N. Karunanithi and Y. C. Sun, "Implementation Considerations for Software Reliability", *Proc. 8th Annual IEEE Soft. Rel. Symp.*, 1990 Jun, pp 6.21-6.30.

[11] Y. K. Malaiya, N. Karunanithi and P. Verma, "Predictability Measures for Software Reliability Models", *Proc. 14th IEEE Int. Computer Software and Applications Conf.*, (COMPSAC 90), 1990 Oct, pp 7-12.

[12] Y. K. Malaiya and P. K. Srimani, Eds, **Software Reliability Models: Theoretical Developments, Evaluation and Applications**, IEEE Computer Society Press, 1990.

[13] K. Matsumoto, K. Inoue, T. Kikuno and K. Torii, "Experimental Evaluation of Software Reliability Growth Models", *IEEE Proc. of FTCS-18*, 1988 Jun, pp 148-153.

[14] P. N. Misra, "Software Reliability Analysis", *IBM Systems Journal.*, vol 22, 1983, pp 262-270.

[15] P. B. Moranda, "Predictions of Software Reliability During Debugging", *Proc. of Annual Reliability and Maintainability Symp.*, Washington, DC, 1975, pp 327-332.

[16] J. D. Musa, "A Theory of Software Reliability and its Application", *IEEE Trans. Software Eng.*, SE-1(3), 1975, pp 312-327.

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

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

[19] M. Ohba, "Software Reliability Analysis Models", *IBM J. Research and Development*, vol 28, 1984 Jul, pp 428-443.

[20] M. L. Shooman, "Probabilistic Models for Software Reliability Prediction", In **Statistical Computer Performance Evaluation** , Academic, New York, 1972, pp 485-502.

[21] N. D. Singpurwalla and R. Soyer, "Assessing (Software) Reliability Growth Using a Random Coefficient Autoregressive Process and Its Ramifications", *IEEE Trans. Software Eng.*, vol SE-11, 1985 Dec, pp 1456-1464.

[22] A. N. Sukert, "Empirical Validation of Three Software Error Prediction Models", *IEEE Trans. Reliability.*, vol R-28, 1979 Aug, pp 199-205.

[23] Y. Tohma, K. Tokunaga, S. Nagase and Y. Murata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution", *IEEE Trans. Software Eng.*, vol 15, 1989 Mar, pp 345-355.

[24] Y. Tohma, H. Yamano, M. Ohba and R. Jacoby, "The Estimation of Parameters of the Hyper-Geometric Distribution and its Application to Software Reliability Growth Model", *Tech Rep. No: 900830, Dept. of Computer Science, Tokyo Institute of Technology.*, 1990.

[25] P. Verma and Y. K. Malaiya, "In Search of the Best Software Reliability Model", *Proc. 7th Annual IEEE Soft. Rel. Symp.*, 1989 May, pp 40-92.

[26] S. Yamada, M. Ohba and S. Osaki, "S-Shaped Reliability Growth Modeling for Software Error Detection", *IEEE Trans. Reliability.*, vol R-32, 1983 Dec, pp 475-478.

[27] K. C. Zinnel, "Using Software Reliability Growth Models to Guide Release Decisions", *Proc. IEEE TCSE Software Reliability Subcommittee Meeting*, 1990 Apr, pp 11.1-11.16.