Predictability of Software-Reliability Models

Yashwant K. Malaiya, Senior Member IEEE
Colorado State University, Fort Collins
Nachimuthu Karunanithi
Bellcore, Morristown
Pradeep Verma
Hewlett-Packard, Cupertino

Key Words — **Model comparison, Predictability measure, Software-reliability growth model.**

Reader Aids —
Purpose: Widen state of art
Special math needed for explanations: Statistics
Special math needed to use results: Same
Results useful to: Software reliability theoreticians, Software managers.

Summary & Conclusions — **Though several software-reliability growth models have been proposed to estimate the reliability growth of a software project, few guidelines exist about which model should be used. We present a 2-component predictability measure that characterizes the long-term predictive capability of a model. Component 1, average error, measures how well a model predicts throughout the testing phase. Component 2, average bias, measures the general tendency to overestimate or underestimate the number of faults. Data-sets for both large and small projects from diverse sources with various initial fault density ranges have been analyzed. Our results show that: i) the Logarithmic model seems to predict well in most data-sets, ii) the Inverse Polynomial model can be used as the next alternative, and iii) the Delayed *S*-Shaped model, which in some data-sets fits well, generally performed poorly. Our statistical analysis also shows that these models have appreciably different predictive capabilities.**

1. INTRODUCTION

Establishing the quality of software systems has become a major challenge 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, and 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 SRGM. A comprehensive survey & classification of software reliability models is in [5,11,17].

All SRGM 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, especially during early phases of testing. This is the time when better predictability is required to estimate: 1) the release date, and 2) additional test effort required. Hence selection of an adequate model can be important for a good estimate of reliability-growth of software systems.

Notation

$\lambda(t)$	failure intensity (fault detection rate), at time t; derivative of $\mu(t)$
$\mu(t)$	mean number of failures in $(0, t)$
λ_i, μ_i	observed value of $[\lambda, \mu]$ 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 data-set j corresponding to $\{t_0, \ldots, t_i\}$
Δ_{ij}^k	$(D_{ij}^k - D_j)/D_j$, prediction error
T_i	time between failures $(i-1)$ & i

Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

Acronyms

SRGM	software-reliability growth model(s)
AE	average error
AB	average bias
IFD	initial fault density (usually in faults/kLOC)
Lo,Me,Hi	[low, medium, high] IFD range
LOC	lines of code (usually in 1000's)
DSSM	Delayed *S*-Shaped model
EXPM	Exponential model
INPM	Inverse Polynomial model
LOGM	Logarithmic model
POWM	Power model
ANOVA	analysis of variance

2. SOFTWARE-RELIABILITY GROWTH MODELS

Five of the most commonly used execution time SRGM are examined here. The most common approach is to use grouped data [9]. The testing duration is divided into several 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 $\lambda(t_F)$ would have fallen below an acceptable threshold.

EXPM [14,15] with its variations is one of the most common. LOGM [16] is one of the more recent models. DSSM [25] is a recent addition to the family of Gamma distribution models. We examined the INPM [8] and the POWM [4]. They are all 2-parameter models. This allows a fair comparison among the models. We believe that these models do represent a sufficiently wide range of presumed behavior. They are all nonhomogeneous Poisson process models with the exception of the inverse-polynomial.

For some models, the parameters have a specific interpretation. We first describe the important steps involved in parameter estimation of LOGM. Since the same steps can be applied for other SRGM, we show only their basic equations. Since the number of data points is not large, we have used least squares estimation in our analyses. The maximum likelihood method performs similarly in this application [17].

2.1 *Logarithmic Model*

LOGM was proposed by Musa & Okumoto [16]. The underlying software failure process is modeled as a logarithmic Poisson process wherein the total number of failures in the system is "infinite in infinite time". The intensity function decreases exponentially with the number of failures. The μ & λ are [17]:

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

$$\lambda(t;\beta) = \beta_0 \cdot \beta_1 / (1 + \beta_1 \cdot t)$$

$$\lambda(\mu;\beta) = \beta_0 \cdot \beta_1 \cdot \exp(-\mu/\beta_0).$$

The square of the sum of the errors is:

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

Notation

r_l actual failure intensity at t_l, calculated from the input data.

Minimizing S gives the least-square estimates of β_0 & β_1. $\beta_0 \cdot \beta_1$ is the failure intensity at time 0. This model belongs to the infinite-fault category [17], and thus the concept of an initial number of faults does not exist.

2.2 *Inverse Polynomial Model*

INPM was proposed by Littlewood & Verrall [8]. It is more general, and is flexible enough so that the user can choose from a variety of reliability growth functions. In our analysis we used a 2-degree polynomial. The main feature of INPM is that the program hazard rate decreases with time and has discontinuities of varying heights at each failure. The μ & λ are [17]:

$$\mu(t;\beta) = 3 \cdot \beta_0 \cdot (Q_1 + Q_2)$$

$$\lambda(t;\beta) = [\beta_0/\sqrt{t^2 + \beta_1}] \cdot (Q_1 - Q_2)$$

$$Q_1 \equiv [t + (t^2 + \beta_1)^{1/2}]^{1/3}$$

$$Q_2 \equiv [t - (t^2 + \beta_1)^{1/2}]^{1/3}$$

Although this is not a popular model, we examined it because it had good predictability for one data-set [17].

2.3 *Exponential Model*

EXPM was proposed by Moranda [14] and reformulated by Musa [15] in terms of execution time. Several models are variations of EXPM [17]. Here the important assumption is that the per-fault hazard rate is a constant. The μ & λ are [17]:

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

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

This is a finite-failures model. β_0 is the initial number of faults; $\beta_0 \cdot \beta_1$ is the initial failure intensity. Techniques exist to estimate β_0 & β_1 empirically [17].

2.4 *Power Model*

POWM was proposed by Crow [4] to estimate reliability of hardware systems during development testing. POWM models the failure events as a nonhomogeneous Poisson process whose failure intensity function is a power function of time. This model can be used to estimate software reliability by controlling the failure intensity range. We have kept $\beta_1 < 1.0$. The μ & λ are [17]:

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

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

This is a infinite-failures model.

2.5 *Delayed S-Shaped Model*

DSSM was proposed by Yamada, Ohba, Osaki [25] and characterizes the software-failure process as a delayed *S*-shaped growth. Software-error detection can be regarded as a learning process in which test-team members improve their familiarity and skill gradually. This is regarded as best of the several *S*-shaped models [25]. The μ & λ are [17]:

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

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

This is a finite-failures model. β_0 is the total number of faults; $1/\beta_1$ 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 & 2 show the typical behavior of these SRGM in terms of μ & λ.

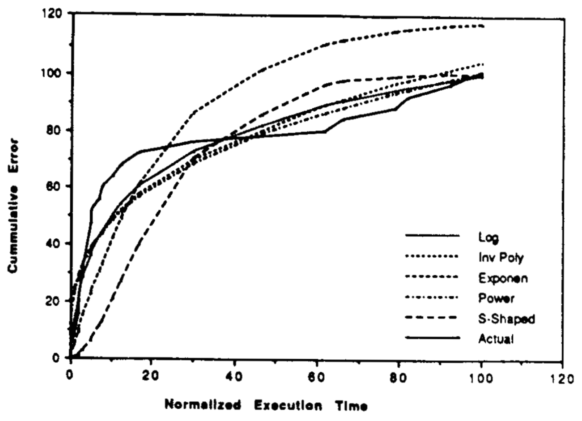

Figure 1. Predicted Cumulative Errors of Models [data-set 4]

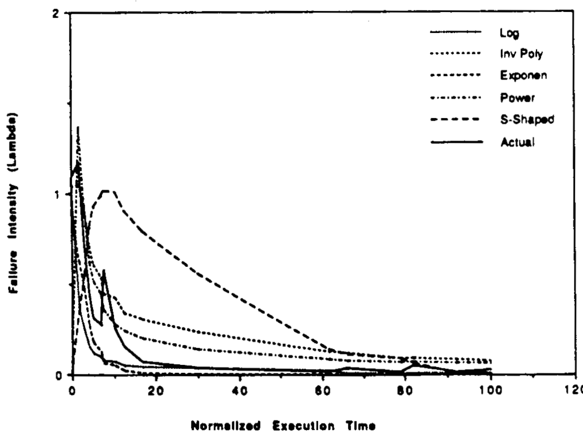

Figure 2. Failure Intensity of Models [data-set 4]

3. A NEW PREDICTABILITY MEASURE

Even though many SRGM are available in the literature, no clear guidelines have been presented for selecting a particular one. Characterization of an SRGM requires applying it to many data-sets and evaluating its predictive capabilities. Applicability of an SRGM can be measured using one of the following approaches:

1. Goodness-of-fit: This is an end-point approach because it can be evaluated only after all the data points are available. The data points are: $\{t_i, \lambda_i: i=0,1,\ldots,n\}$ or equivalently $\{t_i, \mu_i: i=0,1,\ldots,n\}$. After a curve corresponding to a selected M_k is fitted to the data, the deviation between observed and the fitted values is evaluated using a goodness-of-fit test, *eg*, chi-square or Kolmogorov-Smirnov (K-S). The K-S test is considered better than the chi-square test. The goodness-of-fit approach has low computational requirements and is the one used most widely. For example, it was used to compare the exponential, hyperexponential, and *S*-shaped models [12].

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 the earlier. Such a model can have a good overall fit while providing poor predictability in the early phases.

2. Next-Step-Predictability: A partial data-set, say, $\{t_i: i=1,\ldots,(l-1)\}$ is used to predict the T_l. The predicted & observed values of T_l are then compared. This approach has been used in [1,3], and can be used for grouped data. It allows predictability to be measured with a partial data-set, while testing is still in progress. However it measures only short-term predictability.

3. Variable-Term-Predictability: Short-term predictability is an appropriate measure near the end of the test phase. However in practice, it is very important to be able predict the behavior near the end of the test phase by using data from the beginning of the test phase, This approach, used in [10,17] projects the final value of μ at each t_l, using the partial data-set $\{t_i, \mu_i: i=0,\ldots,l\}$. The error in these projections can then be plotted against time. This method requires n curve-fittings for each data-set. The effectiveness of the weighted-parameter estimation has been examined [24] using the same approach. Sukert [21] has empirically validated Jelinski-Moranda, Schick-Wolverton, and modified Schick-Wolverton models using data-sets from four US Dept. of Defense projects; however there were no formal comparisons. □

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

Our 2-component predictability measure consists of Average Error (AE) and Average Bias (AB). AE measures how well a model predicts throughout the test phase. AB measures the general bias of the model. AB can be positive or negative depending on whether the model is prone to overestimation or underestimation. More formally, let —

- the data be grouped into n points $\{t_i, \lambda_i: i=1,\ldots,n\}$
- the specific model be M_k
- data-set be j.

Then D_{ij}^k is a function of $\{M_k; (t_1,\lambda_1),\ldots,(t_i,\lambda_i)\}$.

Taking the variable-term approach [17], we can plot Δ_{ij}^k, $i=1,\ldots,n$, for each model k. The predictability measures are (using the n observation points, i):

$$\mathcal{AE}_j^k = \overline{|\Delta_{ij}^k|},$$

$$\mathcal{AB}_j^k = \overline{\Delta_{ij}^k}.$$

Figure 3 illustrates the use of these measures. The total shaded area under the curve is used for computing AE_j^k, whereas calculating AB_j^k requires use of the appropriate sign.

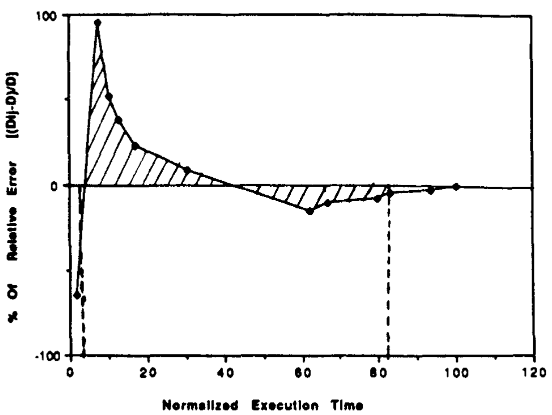

Figure 3. Illustration of Predictability Measures [data-set 4: AE = 33.11, AB = +25.86 t_{lb} = 3.86, t_{ub} = 81.96]

Notation

t_{lb}, t_{ub}	times corresponding to [lower, upper] bound fault densities
m_j	number of points between t_{lb} & t_{ub} for data-set j.

The data-sets correspond to various IFD ranges. For a proper comparison, we trimmed some of the data so that they correspond to one of a few selected ranges. Our lower limits correspond to a point where the relative error is appreciably high. Our upper limits correspond to a point where sufficient faults have been detected (and corrected) and the system has achieved a high reliability. We calculated AB & AE only within these limits which are specific to each data-set (*viz*, using only the m_j observation points, i) and do not depend on the model:

$$\mathrm{AE}_j^k = \overline{|\Delta_{ij}^k|},$$

$$\mathrm{AB}_j^k = \overline{\Delta_{ij}^k}.$$

Our observations show that the omission of end-points does not affect the generality of our conclusions.

The data-sets in table 1 are classified into 3 IFD ranges:

High	10.0	< IFD
Low		IFD < 1.0
Medium		otherwise

High is commonly encountered at the beginning of unit test or the system-test phase [17]. *Medium* corresponds to systems that are either in system-test phase or in some data-sets, in the operational phase. *Low* corresponds to software which has very low IFD, and is usually encountered in the operational phase. These ranges allow us to see if the predictability of an SRGM depends appreciably on IFD. Since the data-sets are for diverse projects from diverse teams, the observations are more useful. If the data-sets are from a single environment, the applicability of the results is less useful.

4. COMPARISON OF MODELS

Our analysis used 18 data-sets collected from a wide variety of software systems [2,12,13,17-20,22,23] 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 real system. Among them, 8 data-sets are from Japanese software projects. Data-sets 3.1, 3.2, 5-9 are calendar-time based while the

TABLE 1
Data-sets Used

Data Sets	Ref	Size (kLOC)	Faults Detected	Software Type	IFD Range (faults/kLOC)	
1.1	[12]	1	27	Class Compiler Project	20.0-5.0	Hi
1.2	”	1	24	”	”	”
1.3	”	1	21	”	”	”
1.4	”	1	27	”	”	”
2	[17]	5.4	136	Real-time Command and Control	”	”
3.1	[18]	40	46	On-line Data Entry	1.00-0.05	Me
3.2	”	1317	328	Database Application Software	0.20-0.05	Lo
3.3	”	35	279	Hardware Control Software	7.0-0.7	Me
4	[20]	180	101	Military Application Software	0.20-0.05	Lo
5	[19]	60	3207	Application Software	13.0-1.3	Hi
6	[22]	870	535	Real-time Control Application	0.5-0.05	Lo
7	[23]	200	481	Monitoring and Real-time Control	2.0-0.05	Me
8	[22]	3.6	55	Railway Interlocking System	3.0-0.05	Me
9	”	90	198	Monitoring and Real-time Control	2.0-0.05	Me
10.1	[2]	10	118	Flight Dynamic Application	10.0-1.0	Hi
10.2	”	22.5	180	Flight Dynamic Application	7.0-0.7	Me
10.3	”	38.5	213	Flight Dynamic Application	5.0-0.5	Me
11	[13]	1000	231	unknown	0.20-0.05	Lo

TABLE 2
Model AE *vs* IFD Range
[The column under a model is its AE. *Mean* is the average AE for all 5 models]

Data-set	LOGM	INPM	EXPM	POWM	DSSM	Mean
			HIGH IFD 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 IFD 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 IFD 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
Model AB *vs* IFD Range
[The column under a model is its AB. *Mean* is the average AB for all 5 models]

Data-set	LOGM	INPM	EXPM	POWM	DSSM	Mean
			HIGH IFD 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 IFD 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 IFD 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

remaining data-sets are execution-time based. The sizes of the source code for data-sets 2, 5, 8 were reported in assembly instructions whereas for the remaining data-sets it was in high-level languages. So we converted the assembly instructions into an equivalent high-level language source code by using a conversion factor of 4 to 1. While this analysis used more data-sets than most other studies, there is a need to collect even more data-sets from diverse sources and repeat the examination. Performance (AE & AB) of these models is shown in tables 2 & 3, and in figures 4 & 5. In table 2, a smaller AE implies a higher accuracy in predictability of a given SRGM. The AB in table 3 indicates whether an SRGM is prone to overprediction or underprediction.

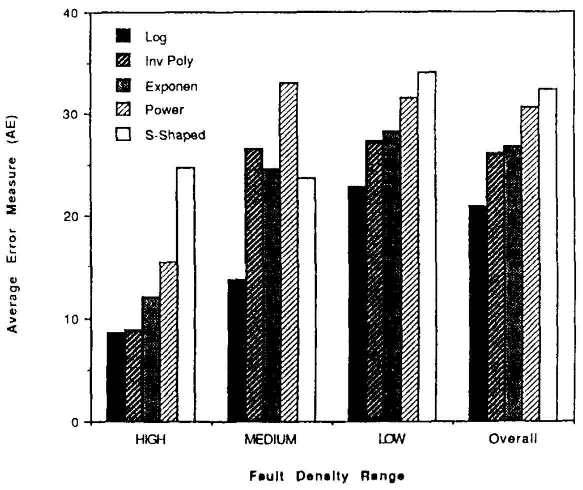

Figure 4. Model AE *vs* IFD Range

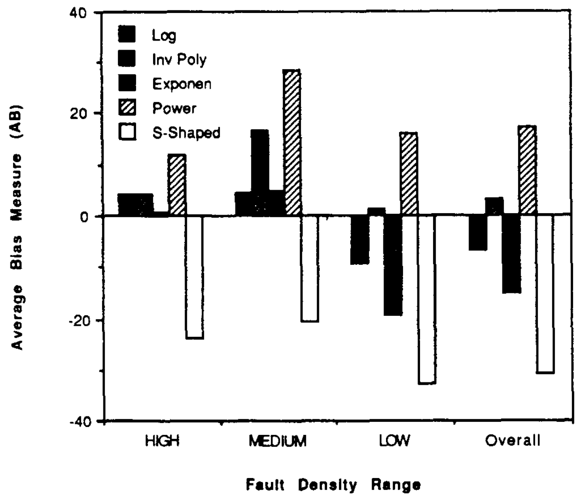

Figure 5. Model AB *vs* IFD Range

4.1 *Comparison with Weights*

This section assumes that the results from the large projects are more important, hence we have included a weighting factor corresponding to the size of the software when averaging. Table 4a represents the weighted-AE and mean weighted-AE of these SRGM. Table 4b represents the weighted-AB of these SRGM. Figure 4 highlights the AE in predictability of SRGM at various IFD ranges across all data-sets. Figure 5 highlights how these SRGM are biased, at various IFD ranges.

TABLE 4a
Model Weighted-AE *vs* IFD Range
[The column under a model is its weighted AE]

IFD	LOGM	INPM	EXPM	POWM	DSSM
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
Mean	20.9	26.0	26.8	30.6	32.4

TABLE 4b
Model Weighted-AB *vs* IFD Range
[The column under a model is its weighted AB]

IFD	LOGM	INPM	EXPM	POWM	DSSM
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
Mean	−6.9	+3.2	−15.2	+17.2	−30.8

LOGM, as shown in table 2, performs relatively well. Though it is not the best predictor in all data-sets, it has projected the remaining faults most accurately in the medium IFD range. Also, in the high & low IFD ranges, its AE are better than those of other SRGM in majority of data-sets. When the best values are predicted by other SRGM in these two ranges, the LOGM AE are often close. As shown in table 3, its AB indicate that LOGM has a mixed bias, sometimes predicting more and sometimes predicting fewer than actual number of faults. The weighted-AE in table 4a and figure 4 suggest that the LOGM performed better than other SRGM. Table 4b indicates that the LOGM exhibited a slight positive bias in high & medium IFD ranges, although the mean AB (over all IFD ranges) showed a slight negative bias. Figure 5 summarizes the behavior of LOGM in terms of AB. This suggests that the LOGM might offer good predictability during various phases, be it testing or field operations.

INPM predicts decently. It has the lowest AE (17.31 & 7.11) for data-sets 1.2 and 2 in the high IFD range, and its AE are in the medium IFD range in most remaining data-sets. Figure 4a shows that INPM has the second best predictive capability in the high & low IFD ranges. From the average AE in figure 4, the INPM is the second best predictor. Table 3 shows that INPM has the tendency of positive bias in over half the data-sets. Table 4b and figure 5 indicate that this SRGM is a consistent overpredictor in all IFD ranges. However, its average AB has the lowest positive value among all SRGM. This might make it suitable for adaptive implementation or for getting a high estimate.

EXPM has AB similar to that of LOGM. Except for data-set 4, where EXPM has the lowest value (15.66), its AE in most remaining data-sets are about in the middle. The AE in table 4a and figure 4 indicate that the EXPM is the third best predictor in all IFD ranges. Except for data-sets 5-7, 9 where it has positive AB, its negative AB lie between the extreme values predicted by other SRGM. In many data-sets, the relatively higher AB of EXPM imply that it might underpredict more, compared to LOGM. Table 4b shows that EXPM has a small positive bias in the high & medium IFD ranges and an appreciable negative bias in the low IFD range. Figure 5 shows that the mean AB is negative.

POWM, as shown in table 2, performs very inconsistently. Though the AE of POWM (16.31, 12.26, 15.29) are the best for data-sets 1.1, 1.3, 11, it has the highest AE in 8 data-sets (1.2, 3.2, 3.3, 4, 6, 8, 9, 10.1). Table 4a and figure 4 show that its AE is the highest among all SRGM in the medium IFD range and the second highest in the low IFD range. Even in the high IFD range its prediction errors are close to the highest values given by the DSSM. Thus POWM might not be a good predictor once the product reaches higher reliability. Table 3 shows that both INPM & POWM have similar AB in all data-sets except 1.3, 10.1, 10.3. The AB of POWM are the highest positive values, as shown in table 4a and figure 5.

DSSM, as shown in table 2, is a poor estimator for many data-sets. Its AE are the highest for data-sets 1.1, 1.3, 1.4, 3.1, 5, 10.1, 10.3, 2, 11. DSSM performs poorly in the high IFD range as shown in table 4a. In the medium IFD range its performance is not appreciably better than that of the worst predictor, POWM. The mean weighted-AE in figure 4 implies that DSSM predicts with the highest error among all SRGM. Its AB in table 3 indicate that it has consistently negative AB across all data-sets, and the highest negative AB for all data-sets except 1.2, 1.3, 8. Its weighted-AB in table 4b and figure 5 show that DSSM is the only SRGM that consistently underestimates in all IFD ranges. DSSM is stable in early phases of testing which explains why some people prefer to use it. Overall it is a weak model. Zinnel [26] has shown that a corrective strategy can improve the performance of this SRGM. This needs to be further investigated.

Table 3 suggests that performance of these SRGM can be affected by the peculiarities of the data-set. For example, data-sets 1.1, 11 forced all models to exhibit negative AB. One approach to overcome this peculiarity can be to use either adaptive prediction [3] or a non-parametric approach [6,7]. Table 4a suggests that all these SRGM, with the exception of DSSM, have decreasing predictive accuracy when IFD is decreased. Further evaluation is needed to verify that this is indeed true. One explanation is that at low failure intensity, the information content in the available data is lower.

4.3 *Comparison without Weights*

Section 4.2 compared the predictive accuracy of SRGM at various IFD ranges using weighted-AE. In order to make our observations stronger, we used ANOVA without assigning weights to the data-sets. We viewed the results as outcomes of a randomized block experiment in which the data-sets were randomly selected and the SRGM were "treatments" applied to each data-set.

We classified the AE into a data-set *vs* SRGM 2-way table and performed an ANOVA; data-sets were the blocks and SRGM the treatments. This approach accounts for the data-set peculiarities which could affect the performance of these competing SRGM.

Since we are also interested in comparing SRGM at various IFD ranges, we performed a variant of the 2-way ANOVA by treating observations as outcomes of an unbalanced nested experiment. Table 5 shows the results of this ANOVA.

TABLE 5
ANOVA Results with Various IFD Ranges

Source	df	MSS	*F* ratio
IFD	2	307.0	
Data-sets within IFD	15	242.2	
SRGM	4	615.6	5.72
IFD *vs* SRGM	8	94.4	0.87
SRGM *vs* Data-sets within IFD#	60	107.6	

*This is used as the reference sum-of-squares, in lieu of any replication.

To find any *s*-significant difference among the SRGM, we used the *F* ratio: (SRGM MSS)/(Reference MSS) = 5.72 with 4 df and 60 df, and is *s*-significant at the 1% level. To find any *s*-significant interaction between the IFD levels and the SRGM, we used the *F* ratio: (IFD *vs* SRGM MSS)/(Reference MSS) = 0.87 with 8 df and 60 df, and is not *s*-significant even at the 50% level. Since the interaction is not *s*-significant the performance of an SRGM relative to the other SRGM is not affected by IFD. The resulting mean AE are shown in table 6 and figure 6.

TABLE 6
Model AE *vs* IFD Range
[The column under a model is its AE]

IFD	LOGM	INPM	EXPM	POWM	DSSM
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
Mean	17.6	24.5	25.2	32.5	32.4

Since the SRGM variations are *s*-significant, the Least *s*-Significant Difference (LSD) procedure can be used to differentiate among them. Using LSD, two SRGM are *s*-significantly

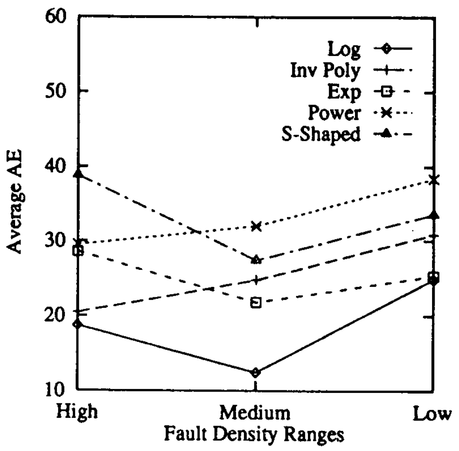

Figure 6. Model AE *vs* IFD Range

different at the 5% level if their means have a difference of at least 6.9 ($T_{5\%,68}=2.0$). If the difference is more than 6.9 then we presume that the SRGM do not have similar predictive accuracy across all data-sets. Thus we conclude, using ">" to denote "*s*-significantly better", that:

{LOGM} > {INPM, EXPM} > {POWM, DSSM}.

This conclusion agrees with our earlier observations based on the weighted analysis. From these graphs we observe:

- the performance of these SRGM can vary at different IFD ranges
- LOGM performs relatively well across various IFD ranges.

However these observations should be further verified using more data-sets.

5. FINAL REMARKS

What we have evaluated is not just the SRGM but rather the combination of our evaluation scheme and SRGM. Thus, whenever our approach is used for evaluating a (new) SRGM the effectiveness of our evaluation scheme should also be considered.

Both calendar-time and execution-time data-sets were used in our analysis. Although one might anticipate that DSSM would have better predictability for calendar-time based data-sets, we did not notice any appreciable difference in performance. Furthermore, our results did not indicate any important difference (nor any *s*-significant difference) in the performance of other SRGM due to the use of either calendar-time or execution-time. This aspect of the problem needs further study.

Some SRGM tend to overestimate or underestimate. INPM & POWM are prone to consistent overprediction whereas the DSSM is prone to consistent underprediction. Some data-sets have peculiarities that can cause most SRGM to have either positive or negative AB. It might be possible to exploit this fact to do an *a priori* or an adaptive correction as shown in [3]. An alternative approach to avoid such biased predictions is to combine at least 2 SRGM with opposite biases.

ACKNOWLEDGMENT

We are pleased to thank the anonymous reviewer who provided many useful comments and suggested the use of ANOVA. This work was supported from a project funded by SDIO/IST and monitored by ONR.

REFERENCES

[1] A. A. Abdel-Ghaly, P. Y. Chan, B. Littlewood, "Evaluation of competing software reliability predictions", *IEEE Trans. Software Engineering*, vol SE-12, 1986 Sep, pp 950-967.

[2] B. M. Anna-Mary, "A study of the Musa reliability model", *MS Thesis*, 1980; Computer Science Dept, University of Maryland.

[3] S. Brocklehurst, P. Y. Chan, B. Littlewood, J. Snell, "Recalibrating software reliability models", *IEEE Trans. Software Engineering*, vol 16, 1990 Apr, pp 458-470.

[4] L. H. Crow, "Reliability for complex repairable systems", *Reliability and Biometry*, 1974, pp 379-410; SIAM.

[5] A. L. Goel, "Software reliability models: Assumptions, limitations and applicability", *IEEE Trans. Software Engineering*, vol SE-11, 1985 Dec, pp 1411-1423.

[6] N. Karunanithi, Y. K. Malaiya, D. Whitley "Prediction of software reliability using neural networks", *Proc. IEEE Int'l Symp. Software Reliability Engineering*, 1991 May, pp 124-130.

[7] N. Karunanithi, D. Whitley, Y. K. Malaiya "Prediction of software reliability using connectionist approaches", *IEEE Trans. Software Engineering*, vol 18, 1992 Jul, pp 563-574

[8] B. Littlewood, J. L. Verrall, "A Bayesian reliability model with a stochastically monotone failure rate", *IEEE Trans. Reliability*, vol R-23, 1974 Jun, pp 108-114.

[9] Y. K. Malaiya, S. Sur, N. Karunanithi, Y. C. Sun, "Implementation considerations for software reliability", *Proc. 8th Ann. IEEE Software Reliability Symp*, 1990 Jun, pp 6.21-6.30.

[10] Y. K. Malaiya, N. Karunanithi, P. Verma, "Predictability measures for software reliability models", *Proc. 14th IEEE Int'l Computer Software & Applications Conf*, (COMPSAC 90), 1990 Oct, pp 7-12.

[11] Y. K. Malaiya, P. K. Srimani (eds), *Software Reliability Models: Theoretical Developments, Evaluation and Applications, 1990;* IEEE Computer Society Press.

[12] K. Matsumoto, K. Inoue, T. Kikuno, K. Torii, "Experimental evaluation of software reliability growth models", *IEEE Proc. FTCS-18*, 1988 Jun, pp 148-153.

[13] P. N. Misra, "Software reliability analysis", *IBM Systems J*, vol 22, 1983, pp 262-270.

[14] P. B. Moranda, "Predictions of software reliability during debugging", *Proc. Ann. Reliability & Maintainability Symp*, 1975, pp 327-332.

[15] J. D. Musa, "A theory of software reliability and its application", *IEEE Trans. Software Engineering*, vol SE-1, 1975 Sep, pp 312-327.

[16] J. D. Musa, K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", *Proc. 7th Int'l Conf. Software Engineering*, 1984, pp 230-238.

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

[18] M. Ohba, "Software reliability analysis models", *IBM J. Research & Development*, vol 28, 1984 Jul, pp 428-443.

[19] M. L. Shooman, "Probabilistic models for software reliability prediction", *Statistical Computer Performance Evaluation*, 1972, pp 485-502; Academic Press.

[20] N. D. Singpurwalla, R. Soyer, "Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications", *IEEE Trans. Software Engineering*, vol SE-11, 1985 Dec, pp 1456-1464.

[21] A. N. Sukert, "Empirical validation of three software error prediction models", *IEEE Trans. Reliability*, vol R-28, 1979 Aug, pp 199-205.

[22] Y. Tohma, K. Tokunaga, S. Nagase, Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution", *IEEE Trans. Software Engineering*, vol 15, 1989 Mar, pp 345-355.

[23] Y. Tohma, H. Yamano, M. Ohba, R. Jacoby, "The estimation of parameters of the hyper-geometric distribution and its application to software reliability growth model", *Tech Rep. No. 900830*, 1990; Dept. Computer Science, Tokyo Institute of Technology.

[24] P. Verma, Y. K. Malaiya, "In search of the best software reliability model", *Proc. 7th Ann. IEEE Software Reliability Symp*, 1989 May, pp 40-92.

[25] S. Yamada, M. Ohba, S. Osaki, "*S*-Shaped reliability growth modeling for software error detection", *IEEE Trans. Reliability*, vol R-32, 1983 Dec, pp 475-478.

[26] K. C. Zinnel, "Using software reliability growth models to guide release decisions", *Proc. IEEE TCSE Software Reliability Subcom. Mtg*, 1990 Apr, pp 11.1-11.16.

AUTHORS

Dr. Yashwant K. Malaiya; Computer Science Dept; Colorado State University; Fort Collins, Colorado 80523 USA.

Yashwant K. Malaiya (S'76,M'78,SM'89) is a Professor in the Department of Computer Science and in the Department of Electrical Engineering at Colorado State University. He has published widely in software & hardware reliability, fault modeling, testing, and testable design. He has also been a consultant for industry. He was the General Chair'n of the 24th Int'l Symp. Microarchitecture and the Program Chair'n of the 5th Int'l Conf. VLSI Design. He has co-edited the IEEE Computer Science Technical Series book, *Software Reliability Models, Theoretical Developments, Evaluation and Applications*. He is the Chair'n of Technical Committee on Microprogramming and Microarchitecture and a Vice-Chair'n of the TCSE subcommittee on software reliability engineering. He received the BSc from Government Degree College, Damoh, the MSc from University of Saugor, and MSc Tech. from BITS, Pilani in India. In 1978 he received the PhD in Electrical Engineering from Utah State University. He was with State University of New York at Binghamton during 1978-82.

Nachimuthu Karunanithi; 2E-378; Bellcore; Morristown, New Jersey 07960 USA.

Nachimuthu Karunanithi received BE (Honors) in Electrical Engineering from P.S.G Tech., Madras University, and ME from Anna University, Madras, in 1982 & 1984. Until 1987 July he was with C-DOT, New Delhi, India, where he participated in the design and development of software for a real-time digital switching system. He was a PhD student in the Department of Computer Science at Colorado State University until 1992 Sep. His research interests are software reliability modeling, neural networks, and genetic algorithms. He is a member of the IEEE TCSE subcommittee on software reliability engineering.

Pradeep Verma; Information Network Division; Hewlett-Packard; Cupertino, California 95014 USA.

Pradeep Verma received a BE in Electrical Engineering from Roorkee University and MTech from IIT Bombay, India in 1984 & 1986. He obtained an MS in Computer Science from Colorado State University in 1989. He is with Hewlett-Packard.

Manuscript TR90-085 received 1990 May 17; revised 1991 March 25; revised 1991 October 25; revised 1992 March 16.

IEEE Log Number 00714 ◄TR►