PREDICTABILITY MEASURES FOR SOFTWARE RELIABILITY MODELS

Yashwant K. Malaiya and Nachimuthu Karunanithi
Computer Science Department
Colorado State University, Fort Collins, CO 80523.
Pradeep Verma
Hewlett-Packard, Information Network Division
19420 Homestead Road, Cupertino, CA 95014.

Abstract

It is critical to be able to achieve an acceptable quality level before a software package is 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, there exist no clear guidelines 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 perform better than others in most cases.

1 INTRODUCTION

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 (or its inverse MTTF). 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 [5,13].

There is evidence to suggest that different models 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.

Here five of the most commonly used fault count models are considered. 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 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 $\{t_i, \lambda_i\}$, or equivalently $\{t_i, \mu_i\}$ 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. Since the number of data points in a data set is often not large we have used the *least squares* method in our experiments. The *maximum likelyhood* method has been found to perform similarly in this application [13].

Logarithmic Model (LOGM): This model was proposed by Musa and Okumoto [12]. Here the underlying software failure process has the characteristics of a logarithmic poisson process. It has an intensity function that decrease exponentially with the number of failures experienced. The mean value function and the failure intensity are [13]:

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

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

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:

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

where r_i is the actual failure intensity at t_i, 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. Since there is no upper limit to the number of faults that would be detected, this model belongs to the infinite-fault category [13], and the concept of an initial number of faults does not exist.

Inverse Polynomial Model (INPM): This model was proposed by Littlewood and Verral [6]. This model is general 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 [13]:

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

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

where,

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

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

Although this is not a popular model, it was chosen for examination because in [13] it has been shown to have good predictability for the data set used for illustration.

Exponential Model (EXPM): This model was originally proposed by Moranda [9] and Musa [10] reformulated it in terms of execution time. Several models can be shown to be variations of this model [13]. 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 finit-failures model. The parameter β_0 represents the initial number of faults and $\beta_0\beta_1$ is the initial failure intensity. Techniques exits to empirically estimate β_0 and β_1 [13].

Power Model (POWM): This model was proposed by Crow [4] 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 [20]. 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 [20]. 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 [23].

2 PREDICTABILITY MEASURES

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 [3] 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 [7].

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-Galey et al [1] and Brocklehurst et al. [2]. 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, is very important. This approach, used by Musa et.al [13], makes projections 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. They have evaluated several models using one data set. The method requires n different curve-fittings for each data set. The effectiveness of the weighted-parameter estimation has been examined by Verma and Malaiya [19] using the same approach.

Sukert [17] 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. Results of several early comparisons have been summarized by Farr [22].

In this paper the variable-term predictability approach has been used to compares some of the major SRGMs. Actual data from a wide spectrum of software projects, corresponding to different initial fault densities has 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 (λ_i, t_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. In terms of functional dependance, it may be specified as

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

Taking the variable-term approach [13], 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 1. 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.

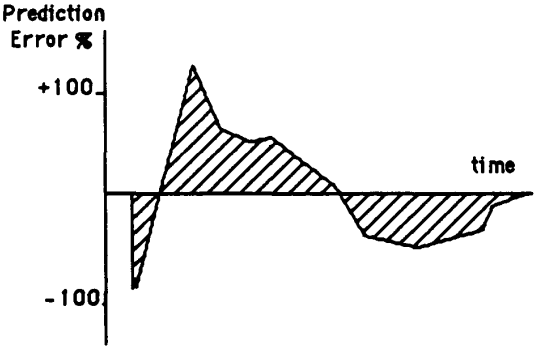
The different data sets used correspond to various fault density ranges. To have a proper comparison, we have trimmed some of the data so that they correspond to one of a few selected ranges. Let t_l and t_u be the time corresponding to lower and upper bound fault densities. Our lower limits correspond 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 observation showed that omission of end-points do not significantly affect the conclusions.

FIGURE 1. Illustration for Predictability Measure



For the purpose of comparison, the data points examined correspond to one of these ranges: (*a*) 20 to 5 faults/KLOC, (*b*) 13.0 to 1.3 faults/KLOC, (*c*) 7.0 to 0.7 faults/KLOC, and (*d*) 0.2 to 0.05 faults/KLOC. The first range is commonly encountered at the beginning of the system test phase or earlier [13]. The last range corresponds to software which has very low fault density to start with. Often operational phase programs have a fault density in this range. Comparison of these two ranges would allow us to see if the predictability of a model has any significant dependence on fault density. For the two intermediate ranges, only limited data set have been used. They have been included only to be able to observe any major anomalies.

3 COMPARISON OF MODELS

In our analysis we have used ten different data sets collected from seven different projects [11,7,14,16,13,8,15] 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. Performances of these models, in terms of AE and AB, are shown in Tables 2 and 3. In Table 2, a smaller value of AE implies a higher accuracy in predictability of a given model, and vice versa. Similarly AB values in Table 3 indicate whether a model is capable of overprediction or underprediction.

The *logarithmic model* (LOGM), as shown in Table 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 four out of ten data sets. Its AE measures of 18.960, 22.354, 8.798, and 8.028 are the lowest for data sets 1.5, 3.2, 3.3, and 5.1. Further observation shows that the LOGM's AEs are not far away from the best values when predicted by other models. This suggests that the LOGM has good adaptability. As shown in Tables 3 its AB values in all cases, except for data sets 4.1 and 5.1, are negative. This implies that the LOGM has some tendency towards underestimation of the number of faults. Even in a case where it overestimated, the over-projected value is not very high compared to that of other models. We observe that LOGM's performance remains superior except for the range of 0.2 to 0.05 fault/KLOC. The averaged AB measure over all fault density ranges is relatively low compared to all other models. Thus the LOGM offers an advantage of consistent predictability at different phases, be it testing or field operations. The LOGM has relatively low bias for different fault density ranges.

TABLE 1. THE DATA SETS USED.

Data Set	Ref.	Code Size Lines/Inst*	Errors found	Software Type	Range f/KLOC
1.1	[7]	1000	27	Compiler Project	20.0–5.0
1.2	"	1000	24	"	"
1.3	"	1000	21	"	"
1.5	"	1000	27	"	"
3.2	[14]	1,317,000	328	PL/I Database Appl.	0.20–0.05
3.3	[14]	35,000	279	Hardware Control	7.0–0.7
4.1	[16]	180,000	101	Military Application	0.20–0.05
5.1	[15]	240,000*	3207	Application Software	13.0–1.3
P1	[13]	21,700*	136	Command & Control	20.0–5.0
P2	[8]	1,000,000	231	Not Known	0.20–0.05

TABLE 2. AVERAGE ERROR OF MODELS(AEs).

Data Set	MODELS					Data Set Average
	Log	Inv Poly	Expon	Power	S-Shap.	
1.1	23.68	28.03	36.32	16.31	44.81	29.83
1.2	36.00	17.31	44.00	65.27	30.83	38.68
1.3	22.21	33.07	31.35	12.26	47.56	29.29
1.5	18.96	36.86	24.23	41.86	47.60	30.28
3.2	22.35	32.35	30.48	36.82	31.69	30.74
3.3	8.79	37.64	13.33	45.49	30.91	27.23
4.1	33.10	47.62	15.66	66.42	31.28	38.82
5.1	8.02	8.53	9.05	13.03	22.73	12.78
P1	10.58	7.11	37.48	34.25	47.60	27.40
P2	26.29	24.23	33.50	15.28	43.15	28.49
Mod. Ave.	21.00	27.28	27.54	34.74	36.01	

TABLE 3. AVERAGE BIAS OF MODELS(ABs).

Data SEt	MODELS					Data Set Average
	Log	Inv Poly	Expon	Power	S-Shap.	
1.1	-23.68	-20.41	-36.32	-2.40	-44.81	-25.52
1.2	-36.00	+17.31	-44.00	+65.27	-11.40	+2.79
1.3	-22.21	+3.11	-31.35	-4.55	-26.31	-16.26
1.5	-9.14	+26.00	-18.21	+37.44	-29.51	+1.31
3.2	-17.99	+6.07	-24.94	+18.71	-28.70	-9.37
3.3	-4.30	+32.67	-12.89	+45.48	-17.39	-8.71
4.1	+25.86	+44.54	-14.76	+61.14	-31.28	+17.10
5.1	+6.33	+4.44	+5.41	+12.57	-21.95	+1.36
P1	-10.03	+5.17	-37.48	+33.56	-47.60	-11.27
P2	-24.88	-19.57	-32.12	-12.20	-43.15	-26.38
Mod. Ave.	-11.63	+9.93	-24.66	+25.50	-27.93	

The *inverse polynomial model* (INPM) seems to perform neither poorly nor very well. The INPM has the lowest AE value of 17.312 and 7.112 for the data sets 1.2 and P1, and in remaining cases its AEs values are in the middle range. Results in Table 3 show that the INPM has the tendency of positive bias. Though the INPM underestimates the number of faults for the data sets 1.1 and P2, its positive bias for the remaining data sets is not very strong. This suggests that the INPM predicts relatively more accurately in high fault density ranges. For example, in the ranges of 20.0 to 5.0 faults/KLOC, and 13.0 to 1.3 faults/KLOC its AE values are the second lowest, where as at lower fault density ranges of 7.0 to 0.7 faults/KLOC, and 0.2 to 0.05 fault/KLOC its values are very high. It can be seen that the INPM exhibits overprediction in all fault density ranges.

The *exponential model's* behavior is somewhat similar to that of LOGM. Except for data set 4.1, where it has the

lowest value of 15.660, its AE values in most of the remaining cases suggest that the EXPM follows a middle course. But it has the maximum AE of 44.00 for data set 1.2, and the second worst value for data sets 1.1, P1, and P2. Except for data set 4.1, where it has a +ve value of 5.415, its -ve AB values lie between the extreme values predicted by other models. Its AB value of -44.00 and -31.355 for the data sets 1.2 and 1.3 are the worst. In many of the cases, EXPM's relatively higher magnitude of AB values suggests that the EXPM may under-project significantly compared to the LOGM. The AE average shows good predictability in 0.2–0.05 f/KLOC range but poor predictability in the range of 20.0 to 5.0 faults/KLOC. EXPM's bias at the highest fault density range is very high.

The *power model*, as shown in Table 2, seems to perform highly inconsistently. Though the POWM's AE measure of 16.314, 12.260, and 15.286 are the best for data sets 1.1, 1.3, and P2, it has the highest AEs in four cases (data set: 1.2, 3.2, 3.3, and 4.1). A comparison in Table 3 shows that both the INPM and the POWM seems to perform in a similar manner in all cases except for data set 1.3. But POWM's AB values are the highest +ve values for data sets 1.2, 1.5 through P1. Its average AE values in ranges 7.0 – 0.7 faults/KLOC, and 0.2 – 0.05 faults/KLOC are very high compared to other models. Even in higher fault density ranges its prediction errors are close to the highest values of given by DSSM. This implies that this model may not be a good predictor once the product reaches higher reliability. Values indicate that the POWM and INPM have similar biases in all fault density ranges. But the POWM seems to over-project more than the INPM.

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 in case of the following data sets: 1.1, 1.3, 1.5, 5.1, P1, and P2. 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 and 1.3. Hence one has to be cautious in releasing a product when DSSM is used for reliability prediction. The delayed S-Shaped model performs consistently poorly in high fault density ranges. Also in low fault density ranges its performance is not significantly better than that of the worst predictor, the POWM. AB values show that the DSSM is the only model that consistently underestimates in all fault density ranges.

The last column of Table 3 suggests that performance of these models may sometimes be affected by the peculiarities of some data sets. For example, data sets 1.1 and P2 forced all models to exhibit negative bias in their estimate. One possible approach to overcome this can be to use an adaptive prediction approach.

4 OBSERVATIONS

We have evaluated *average error* and *average bias* for different models for data from different fault density ranges. Our results seem to support Musa's observation [13] that the logarithmic model appears to have good predictability in most cases. However at very low fault densities, the exponential model may be slightly better. The delayed S-Shaped model, which in some cases have been shown to have good fit, generally performed poorly. Similar results have been found using a different evaluation scheme [18]. Zinnel [21] has shown that a significant correction can improve the predictability of the delayed S-Shaped model. This needs to be further investigated.

Some data sets may have peculiarities that may cause most models to have either positive or negative bias. It may be possible to use this fact to do a preset or adaptive correction. Another possible approach may be to combine models with opposite biases.

5 Acknowledgement

This work was partly supported by an SDIO/IST contract monitored by ONR.

References

[1] A.A. Abdel-Ghaly, P.Y. Chan, and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," *IEEE Trans. Reliability.*, Vol SE-12, No-9, 1986 Sep, pp 950-967.

[2] S. Brocklehurst, P.Y Chan, B. Littlewood, and J. Snell, "Recalibrating Software Reliability Models," *IEEE Trans. Software Eng.*, Vol 16,No 9, 1990 Apr, pp 458-470.

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

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

[5] A.L. Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," *IEEE Trans. Software Engg.*, Vol SE-11, No 12, 1985 Dec, pp 1411-1423.

[6] B. Littlewood, and J.L. Verral, "A Bayesian reliability model with a Stochastically monotone failure rate," *IEEE Trans. Reliability.*, Vol R-23,No 2, 1974, pp 108-114.

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

[8] P.N. Misra, "Software Reliability Analysis," *IBM Systems Journal.*, Vol 22,No 3, 1983, pp 262-270.

[9] P.B. Moranda, "Predictions of Software Reliability during debugging," *Proceedings of Annual Reliability and Maintainability Symposium*, Washington, DC, 1975, pp 327-332.

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

[11] J.D. Musa, "Validity of execution-time theory of software reliability," *IEEE Trans. Reliability.*, Vol R-28,No 3, 1979 Aug, pp 181-191.

[12] J.D. Musa, and K. Okumoto, "A logarithmic poisson execution time model for software reliability measurement," *Proc. 7th Int. Conf. on Software Engg.*, 1984, pp 230-238.

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

[14] M. Ohba, "Software reliability analysis models," *IBM J. RES. DEVELOP*, Vol 28,No 4, 1984 July, pp 428-443.

[15] M.L. Shooman, "Probabilistic models for software reliability prediction," In **Statistical Computer Performance Evaluation** , Academic, New York, 1972, pp 485-502.

[16] N.D. Singpurwalla and R. Soyer, "Assessing (Software) Reliability Growth Using a Random Coefficient Autoregressive Process and Its Ramifications," *IEEE Trans. Software Engg.*, Vol SE-11, No 12, 1985 Dec, pp 1456-1464.

[17] A.N. Sukert, "Empirical Validation of Three Software Error Prediction Models," *IEEE Trans. Reliability.*, Vol R-28, No-3, 1979 Aug, pp 199-205.

[18] Y.C. Sun, "Computational Aspects of Software Reliability Model Applications," *M.S Project Report, CS Department, Colorado State University.*, 1990.

[19] P. Verma, and Y.K. Malaiya, "In search of the best software reliability model," *Proc. 7th Annual IEEE Soft. Rel. Symposium*, 1989 May, pp 40-92.

[20] S. Yamda, 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.

[21] 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.

[22] W.H. Farr, **A Survey of Software Reliability Modeling and Estimation**, Naval Surface Weapons Center, 1983, pp. 5.1-5.4

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