An Experiment in Determining Software Reliability Model Applicability

Allen P. Nikora
Jet Propulsion Laboratory/
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Mail Stop 264-805
bignuke@ganymede.jpl.nasa.gov

Michael R. Lyu
Distributed Software Research Department
AT&T Bell Laboratories
Room 2B132
600 Mountain Avenue
Murray Hill, NJ 07974
lyu@research.att.com

Abstract

Most reported experience with software reliability models is from a project's testing phases, during which researchers have little control over the failure data. Since failure data can be noisy and distorted, reported procedures for determining model applicability may be incomplete. To gain additional insight into this problem, we generated forty sets of data by drawing samples from two distributions, which were used as inputs to six different software reliability models. We used several different methods to analyze the applicability of the models. We expected that a model would perform the best on the data sets created to comply with the model's assumptions, but initially found that this was not always the case. More detailed examination showed that a model using a data set created to satisfy its assumptions tended to have better prequential likelihood, bias, and bias trend measures, although the Kolmogorov-Smirnov test might not be a reliable indicator of the best model. These results indicate that more than one measure should be used to determine model applicability, and that for greater accuracy they be evaluated in sequence rather than simultaneously.

1. Introduction

Most of the reports on experience with software reliability models have made use of data from actual software development efforts [8, 11]. These reports have been useful in helping practitioners determine the behavior the models will exhibit under actual test and operational conditions. However, the failure data from actual projects is influenced by process and product characteristics that are not taken into account in most models. The shape of a failure intensity curve may be influenced by the complexity of the software under test, the number of testers available, the skill level of the testing staff, and the type of testing being performed (e.g. functional testing, path testing, data coverage testing). These data are also subject to uncertainty and distortion. The complexity of real world failure data may obscure properties of software reliability models that might be revealed by executing the models on simpler data sets. We have created sets of interfailure times by generating sequences according to the distributions for two of the more-widely used software reliability models, and have executed six models on each data set. The results of this activity have suggested ways in which existing methods might be most effectively used in choosing the most appropriate model. In the remaining sections we discuss the following items:

1. The way in which the data sets were created, and the way the models were executed using these data sets.

2. The results of the experiment, where rankingis of the models with respect to various applicability criteria are analyzed.

1071-9458/95 $4.00 ©1995 IEEE

3. A discussion of the results, and recommendations for further work.

2. Method

In creating the data sets, we generated 40 sequences of interfailure times according to the method discussed in Section 12.3 of [13]. The first 20 sequences were created by drawing random samples from an exponential distribution, while the remaining 20 were created to simulate sequences of interfailure times for a logarithmic Nonhomogeneous Poisson Process. To generate data simulating a logarithmic Nonhomogeneous Poisson Process, the following was done for each sequence:

1. Generate a sequence of random numbers, z_i, uniformly distributed in the interval (0, 1).

2. For each z_i, compute an exponential random value $u_i = -\ln(1 - z_i)$. This represents the i'th failure interval for a stationary Poisson process with a rate of 1.

3. Set $v_i = \sum_{j=1}^{i} u_i$. This represents the i'th failure time for the Poisson process.

4. Convert the failure time for the stationary Poisson process to the failure time for the logarithmic Poisson process, w_i:
$$w_i = \mu^{-1}(v_i)$$
where μ is the mean value function for the logarithmic Poisson model:
$$\mu(\tau) = (1/\Theta)\ln(\lambda_0\theta\tau + 1)$$
λ_0 = initial failure intensity
Θ = failure intensity decay parameter
τ = total elapsed execution time

5. Convert the logarithmic Poisson failure times to failure intervals by taking the differences between successive values of w_i.

A similar process was used to generate the sequences simulating data from an exponential Nonhomogeneous Poisson Process.

Six of the better-known software reliability models were then run on the sequences using maximum likelihood parameter estimation. The six models and their hazard rates or mean value functions are given below. Detailed descriptions may be found in [2], [7], and [13].

1. **Geometric Model** - hazard rate z(t) for time "t" between the i-1'th and the i'th failure is $z_0\Theta^{i-1}$, where z_0 is the initial hazard rate, and Θ is the decay constant.

2. **Jelinski-Moranda Model** - hazard rate is :
$$z(t) = K(\frac{E_T}{I_T} - \epsilon_C t)$$
where $z(\tau)$ represents the hazard rate at failure time τ, K is the proportionality constant, E_T is the number of errors initially in the program, IT is the number of machine instructions in the program, and ϵ_C is the cumulative number of failures removed in the interval $[0, \tau]$.

3. **Littlewood-Verrall Model (quadratic form)** - the hazard rate for the quadratic form is:
$$z(t_i) = \frac{\alpha}{t_i + \beta_0 + \beta_1 i^2}$$
where α, β_0, and β_1 are parameters of the model, i represents the failure number, and t_i is the time between the i-1'th and i'th failures.

4. **Musa Basic Model** - the mean value function for this model is:
$$\mu(t) = \nu_0(1 - \exp(-\frac{\lambda_0}{\nu_0}t))$$
where $\mu(t)$ represents the mean number of failures at time t, λ_0 represents the initial failure

intensity, and v_0 represents the estimated total number of failures that would be observed over an unlimited amount of execution time.

5. **Musa-Okumoto Model** - the mean value function is given at the start of this section.

6. **Nonhomogeneous Poisson Process (NHPP) Model** - the mean value function for this model is given by the following equation:

$$\mu(t) = a(1 - \exp(-bt))$$

where $\mu(t)$ represents the mean number of failures at time t, a is the estimated total number of failures that would be observed over an unlimited amount of execution time, and b is the intensity decay parameter.

The tool CASRE [11] was used to run the models. CASRE is a software reliability modeling tool implemented in a Microsoft Windows environment. The core modeling capabilities of this tool are the libraries originally implemented for version 5 of the software reliability modeling tool SMERFS [6]. The object code for these libraries was linked into the executable CASRE module. The command interface is a set of pull-down menus that make it easy to navigate through the functional areas of the tool. Input data and model results are displayed both as text and as high-resolution graphics that were designed to be easy for non-specialists to interpret. Model results can be written to a text file which can be brought into a spreadsheet, database, or statistical analysis package for further analysis.

An array of estimated Mean Times To Failure (MTTF) was then generated, using the parameter estimates obtained after processing the last observation in each sequence. The Kolmogorov-Smirnov goodness of fit test was used to determine the goodness of fit of the model results to the input data set. To evaluate model applicability, the prequential likelihood, model bias, and bias trend were computed [3]. A brief description of these three criteria is given below:

1. **Prequential Likelihood** - although similar in form to the likelihood function used for maximum likelihood estimation, this function is not used to estimate model parameters. Rather, the parameter estimates and actual observed failure times are used in this function to compute a value that can be used to determine how much more likely it is that one model will produce accurate estimates than another model. This likelihood is given by the value of the ratio of the prequential likelihoods for the two models being compared.

2. **Model Bias** - the estimated probability of failure for each failure interval is used to determine the extent to which a model introduces bias into its estimates. If a model is biased, it can be optimistically biased (estimates of MTTFs are higher than what is actually observed), or it can be pessimistically biased. The cumulative distribution function (cdf) for the estimated failure probabilities is compared to the cdf for iid random variables in the interval (0,1) using the Kolmogorov-Smirnov (KS) test. The KS test statistic reveals the extent of model bias.

3. **Model Bias Trend** - this measure uses the estimated probability of failure for each failure interval to determine whether model bias changes over time. A model may be optimistically biased during the early stages of testing, while it may become pessimistically biased during the later stages. The analysis is similar to that for model bias, except that the estimated failure probabilities are transformed in a way that preserves temporal information.

The models were then ranked with respect to all four criteria. Model noise was not included in the criteria because, unlike the other criteria, it provides no absolute indicator of how well a model performs, nor does it necessarily measure how well one model performs compared with other models.

3. Results

The results of this experiment are summarized in Tables 1-8 below. The first two tables summarize the performance of all models across the input sequences. Table 1 summarizes the performance of all models across the 20 sequences drawn from an exponential distribution, while Table 2 summarizes the performance of all models across the 20 sequences drawn from the logarithmic Poisson distribution. Tables 1 and 2 are interpreted as follows.

1. Even-numbered columns rank the model across all inputs of a specific type with respect to prequential likelihood, model bias, model trend, and the Kolmogorov-Smirnov goodness-of-fit test. For each criterion, the ranks for all input data sets of a given type are summed and entered into the column. Column 1 gives the sum of the overall ranks of each model. The overall rank is computed by equally weighting the individual ranks according to prequential likelihood, model bias, model bias trend, and goodness of fit (KS test).
2. Odd-numbered columns after the first column give the standard deviation for the model rankings summed in the preceding column. For instance, column 2 gives the standard deviation of the overall ranking across all input data sets.

Note that for the exponential data inputs, the NHPP model is ranked low compared to the Musa Basic and Jelinski-Moranda models. This is because the NHPP model was often unable to run to completion. At some point in the data set, the parameter estimates would not converge - this would prevent the computation of values for the KS test, prequential likelihood, model bias, and model bias trend. The model would be ranked last in this case. The same situation is seen for the model results using the simulated logarithmic NHPP data. In this case, the Jelinski-Moranda, the Musa Basic, and the NHPP models were frequently tied for last place because convergent parameter estimates could not be made.

This information is shown in Figures 1 and 2 below. These figures also show the mean ranks according to prequential likelihood, model bias, model bias trend, and goodness of fit.

Tables 3 and 4 summarize the ranking frequencies for each model for a particular input type. For each model that was run using the 20 sets of data drawn from an exponential distribution, Table 3 gives the number of times that model was assigned a particular rank. Table 4 is interpreted the same way, except that the input data to the models was the 20 sequences intended to be representative of a logarithmic Nonhomogeneous Poisson Process.

We found that traditional goodness-of-fit tests (in this case, the Kolmogorov-Smirnov test) do not seem to be the best way of identifying the most appropriate model. For instance, we would expect that the Jelinski-Moranda, the Musa Basic, of the NHPP model would perform best on the data sequences generated by drawing random samples from an exponential distribution. However, Table 5 below indicates that collectively, these models do not rank first according to this measure as often as the Geometric, the Littlewood-Verrall, and the Musa-Okumoto. We see the same phenomenon in the data sets created to simulate a logarithmic Nonhomogeneous Poisson process. In Table 5 below, note that the Littlewood-Verrall model ranks first 12 times, while the Musa-Okumoto model, which we would expect to perform the best, ranks first only three times. Figures 3 and 4 show this information in the form of a 3-D plot.

Looking at the other methods of ranking the models, we find that they provide results much closer to what we might expect. Specifically, the Jelinski-Moranda, Musa Basic, and NHPP models are favored when using the sequences drawn from an exponential distribution as input, while the Musa-Okumoto model is favored when using data simulating a logarithmic Nonhomogeneous Poisson Process as input. This is shown in Tables 6-8 on the following pages.

Model Name	Sum of Overall Rank	Std Dev	Sum of Preqnt'l Likelihood Rank	Std Dev	Sum of Bias Rank	Std Dev	Sum of Bias Trend Rank	Std Dev	Sum of KS Rank	Std Dev
Geometric	74	1.30182	79	1.23438	80	1.29777	77	1.22582	63	1.84320
Jelinski-Moranda	55	1.88833	47	1.84320	46	1.65752	60	1.48678	62	1.65116
Littlewood-Verrall	79	1.63755	77	2.03328	80	1.55597	98	1.55259	72	1.75919
Musa Basic	40	1.02598	46	1.12858	45	0.85070	38	1.41049	59	1.50350
Musa Okumoto	73	1.13671	79	0.88704	99	1.31689	59	0.94451	69	1.39454
NHPP	84	2.14231	83	2.08440	77	2.23077	81	2.08945	99	1.79106

Table 1 - Summary of model rankings - data drawn from an exponential distribution

Model Name	Sum of Overall Rank	Std Dev	Sum of Preqnt'l Likelihood Rank	Std Dev	Sum of Bias Rank	Std Dev	Sum of Bias Trend Rank	Std Dev	Sum of KS Rank	Std Dev
Geometric	39	0.88704	31	0.60481	35	0.44426	49	0.68633	45	0.78640
Jelinski-Moranda	109	0.99868	110	0.88852	110	0.88852	106	1.38031	105	1.33278
Littlewood-Verrall	54	1.08094	62	0.71818	63	0.67082	42	1.44732	37	1.34849
Musa Basic	118	0.44721	119	0.22361	119	0.22361	118	0.44721	119	0.22361
Musa Okumoto	21	0.22361	30	0.51299	25	0.44426	36	0.61559	44	0.69585
NHPP	118	0.44721	118	0.44721	118	0.44721	119	0.22361	120	0.00000

Table 2 - Summary of model rankings - data representative of logarithmic NHPP model

Model Name	Times Ranked First	Times Ranked Second	Times Ranked Third	Times Ranked Fourth	Times Ranked Fifth	Times Ranked Sixth
Geometric	2	1	4	8	4	1
Jelinski-Moranda	7	4	4	1	0	4
Littlewood-Verrall	2	1	6	3	3	5
Musa Basic	8	6	4	2	0	0
Musa Okumoto	1	2	5	7	5	0
NHPP	4	2	2	0	2	10

Table 3 - Overall ranking frequency - exponential NHPP inputs

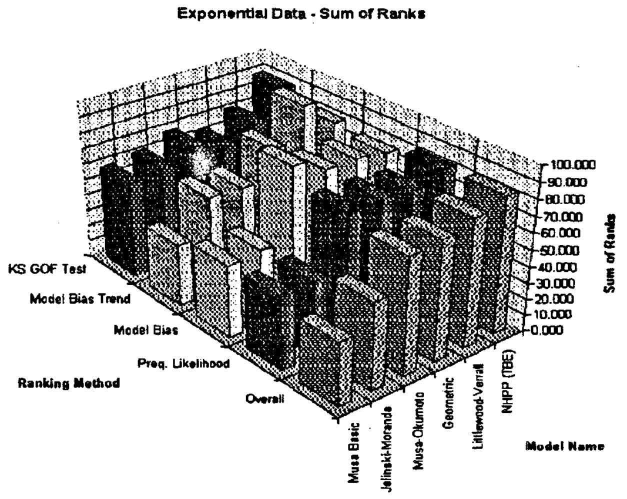

Figure 1 - Sum of Model Rankings - Exponential NHPP Inputs

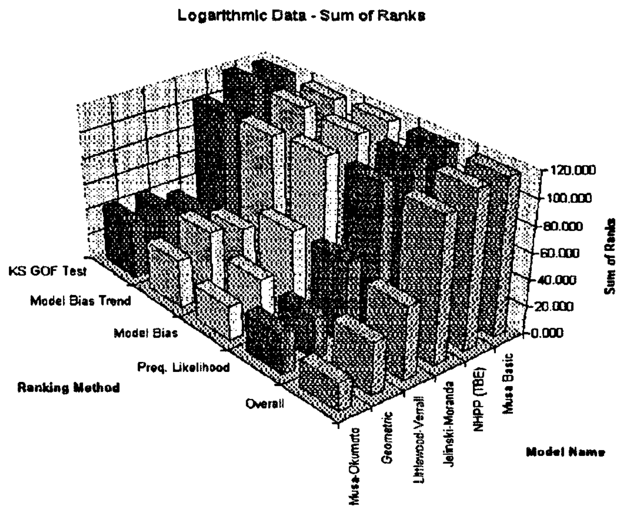

Figure 2 - Sum of Model Rankings - Logarithmic NHPP Inputs

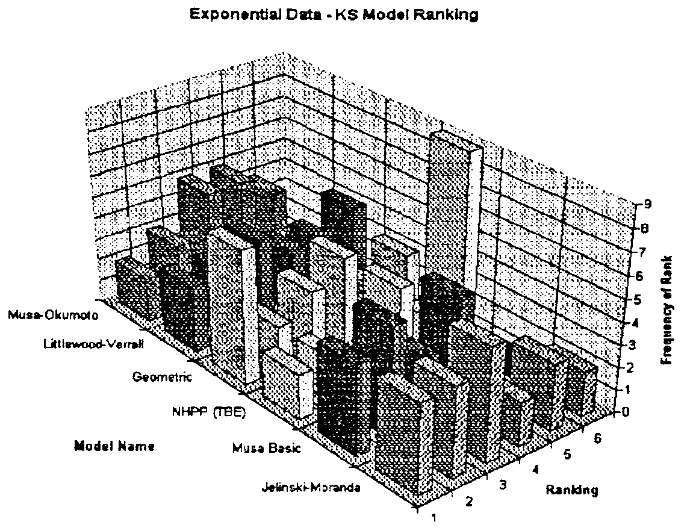

Figure 3 - KS Test Model Ranking - Exponential NHPP Inputs

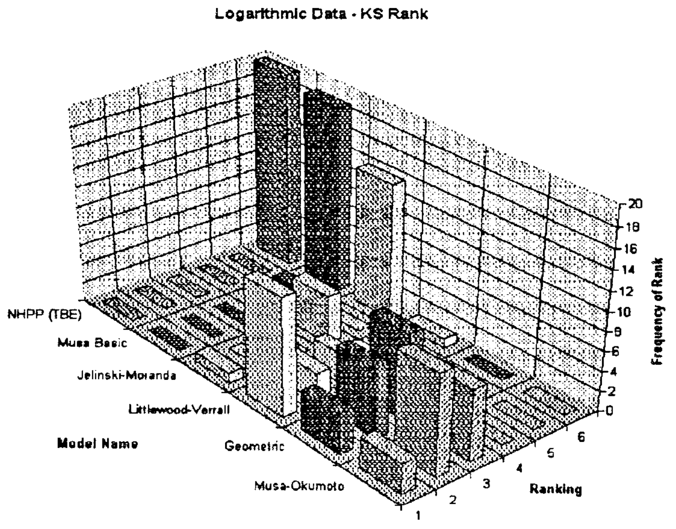

Figure 4 - KS Test Model Ranking - Logarithmic NHPP Inputs

Model Name	Times Rnk'd 1st	Times Rnk'd 2	Times Rnk'd 3rd	Times Rnk'd 4th	Times Rnk'd 5th	Times Rnk'd 6
Geometric	8	5	7	0	0	0
Jelinski-Moranda	0	0	1	4	0	15
Littlewood-Verrall	2	6	10	1	0	1
Musa Basic	0	0	0	1	0	19
Musa Okumoto	19	1	0	0	0	0
NHPP	0	0	0	1	0	19

Table 4 - Overall ranking frequency - data representative of logarithmic NHPP model

Model Name	Times Rnk'd 1st	Times Rnk'd 2	Times Rnk'd 3rd	Times Rnk'd 4th	Times Rnk'd 5th	Times Rnk'd 6
Geometric	(6,4)	(2,7)	(3,9)	(4,0)	(2,0)	(3,0)
Jelinski-Moranda	(4,1)	(4,0)	(5,0)	(2,5)	(3,0)	(2,14)
Littlewood-Verrall	(3,12)	(3,3)	(4,3)	(3,1)	(3,0)	(4,1)
Musa Basic	(4,0)	(5,0)	(3,0)	(5,0)	(2,1)	(1,19)
Musa Okumoto	(2,3)	(3,10)	(5,7)	(5,0)	(4,0)	(1,0)
NHPP	(2,0)	(2,0)	(1,0)	(4,0)	(2,0)	(9,20)

Table 5 - KS Test ranking frequency - (exponential NHPP inputs, logarithmic NHPP inputs)

Model Name	Times Rnk'd 1st	Times Rnk'd 2	Times Rnk'd 3rd	Times Rnk'd 4th	Times Rnk'd 5th	Times Rnk'd 6
Geometric	(1,10)	(3,9)	(0,1)	(8,0)	(8,0)	(0,0)
Jelinski-Moranda	(11,0)	(1,1)	(4,18)	(1,0)	(0,0)	(3,1)
Littlewood-Verrall	(4,0)	(1,0)	(6,0)	(0,5)	(1,0)	(8,15)
Musa Basic	(4,0)	(11,0)	(1,0)	(3,0)	(1,1)	(0,19)
Musa Okumoto	(0,10)	(1,10)	(5,0)	(8,0)	(6,0)	(0,0)
NHPP	(3,0)	(4,0)	(0,0)	(3,1)	(0,0)	(10,19)

Table 6 - Prequential Likelihood ranking frequency - (expnt'l NHPP inputs, logarithmic NHPP inputs)

Model Name	Times Rnk'd 1st	Times Rnk'd 2	Times Rnk'd 3rd	Times Rnk'd 4th	Times Rnk'd 5th	Times Rnk'd 6
Geometric	(2,5)	(1,15)	(1,0)	(7,0)	(9,0)	(0,0)
Jelinski-Moranda	(10,0)	(2,0)	(4,0)	(2,5)	(0,0)	(2,15)
Littlewood-Verrall	(1,0)	(1,0)	(8,19)	(3,0)	(1,0)	(6,1)
Musa Basic	(3,0)	(11,0)	(4,0)	(2,0)	(0,1)	(0,19)
Musa Okumoto	(1,15)	(1,5)	(1,0)	(6,0)	(7,0)	(4,0)
NHPP	(5,0)	(3,0)	(1,0)	(1,1)	(1,0)	(9,19)

Table 7 - Model Bias ranking frequency - (exponential NHPP inputs, logarithmic NHPP inputs)

Model Name	Times Rnk'd 1st	Times Rnk'd 2	Times Rnk'd 3rd	Times Rnk'd 4th	Times Rnk'd 5th	Times Rnk'd 6
Geometric	(1,2	(3,7)	(1,11)	(8,0)	(7,0)	(0,0)
Jelinski-Moranda	(2,1)	(7,0)	(6,1)	(1,3)	(2,0)	(2,15)
Littlewood-Verrall	(2,11)	(0,1)	(1,5)	(1,2)	(7,0)	(9,1)
Musa Basic	(12,0)	(3,0)	(2,0)	(2,1)	(0,0)	(1,19)
Musa Okumoto	(2,6)	(3,12)	(9,2)	(6,0)	(0,0)	(0,0)
NHPP	(4,0)	(2,0)	(2,0)	(2,0)	(1,1)	(9,19)

Table 8 - Model Bias Trend ranking frequency - (exponential NHPP inputs, logarithmic NHPP inputs)

4. Discussion

Although the data sets used as inputs to the models were created to favor either the exponential or logarithmic NHPP models, in a few cases, other models were favored over the ones that were expected to be chosen. Along with earlier work in this area [3], this demonstrates that selection of the most appropriate model for a testing effort must continue after testing has started and model application has begun. Indiscriminate application of statistical methods can result in not choosing the most appropriate model on which to base reliability forecasts. Although a one-step ranking of the models with respect to prequential likelihood, bias, bias trend, and goodness of fit can provide a good idea of the most appropriate model in many cases, there may still be times for which a less appropriate model might be chosen. Part of the problem seems to be that the goodness-of-fit test is not sensitive enough to make fine distinctions among models. It is perhaps better suited to serve as a preliminary screening, rejecting models that do not fit to a pre-specified significance level. For those models that do fit the data to the specified level, successive application of the other three methods can provide a better idea of which model is more appropriate. We recommend following the steps below to choose the most appropriate model.

1. Apply a goodness of fit test to determine if the model results fit the input data to a specified significance level.
2. If more than one set of results are a good fit:
    a. Choose the most appropriate model(s) based on the prequential likelihood.
    b. In the event of a tie, use the model bias, then model bias trend to break the tie.
    c. Use techniques, such as forming linear combinations of model results [9, 10, 11] or model recalibration [4, 5, 9], to increase prediction accuracy. Reports on the use of these methods indicate that they can be used to significantly increase the models' predictive accuracy.
3. If only one model provides a good fit to the data, choose that model.
4. If no models provide a good fit to the data:
    a. Choose the most appropriate model(s) based on the prequential likelihood.
    b. Use the linear combination and re-calibration techniques mentioned above to increase prediction accuracy.
    c. Apply the goodness of fit test to the adjusted model results to identify those that are a good fit to the data.

5. Conclusion

The selection of the most appropriate software reliability model is a process that continues throughout the testing phase. Even if the characteristics of the testing process are well-known for a particular development effort, this is no guarantee that the model whose assumptions appear to best match these characteristics will be the most appropriate model. A staged application of the applicability criteria previously discussed appears to be the best way of selecting the most appropriate model. Reliance on a single measure to choose the most appropriate model can lead to making an incorrect choice. A weighted ranking scheme involving several criteria can reduce the chances of making an incorrect selection, but the risk of choosing an inappropriate model can still be greater than using the multiple opportunities of a staged ranking process to eliminate less appropriate choices.

We intend to investigate and evaluate model recalibration and linear combination techniques using various distributions of interfailure times. We are also planning to examine the behavior of the models with respect to other interfailure time distributions that might be encountered. Further work on identifying the most appropriate model for a development effort is needed. Although prequential likelihood and model bias computation have proven to be useful methods, other approaches, such as the Akaike information criterion [1], should be studied further.

In addition, more work is needed in relating product and development process characteristics to appropriate models. Although it seems to be the case that there is no way to determine with certainty which model is the most appropriate for a particular effort [3], there may be ways of using process and product measures to guide the selection of models that are likely to produce valid predictions. Alternatively, these measures may be incorporated into the model directly so as to better describe the fault detection and removal process [12].

Acknowledgement

The work performed in this paper was partly perfomed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The enhancement and maintenance of CASRE is being supported by the US Air Force Operational Test and Evaluation Center.

References

1 H. Akaike, "A New Look at Statistical Model Identification," IEEE Transactions on Automatic Control, vol. AC-19, pp. 716-723, 1974.

2 ANSI/AIAA R-013-1992, "Recommended Practice for Software Reliability", sponsored by American Institute of Aeronautics and Astronautics, February, 1993.

3 A. A. Abdel-Ghaly, P. Y. Chan, B. Littlewood, "Evaluation of Competing Software Reliability Predictions," IEEE Transactions on Software Engineering, vol. SE-12, pp.950-967, September, 1986.

4 S. Brocklehurst, P. Y. Chan, B. Littlewood, J. Snell, "Recalibrating Software Reliability Models," IEEE Transactions on Software Engineering, vol, SE-16, no. 4, pp. 458-470, April, 1990.

5 S. Brocklehurst, B. Littlewood, "Techniques for Prediction Analysis and Recalibration", Chapter 4 in Handbook of Software Reliability Engineering, M. R. Lyu (ed.), IEEE and McGraw-Hill, New York, 1995

6 W. H. Farr, "Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide", Revision 3, September, 1993

7 W. H. Farr, "Software Reliability Modeling Survey", Chapter 3 in Handbook of Software Reliability Engineering, M. R. Lyu (ed.), IEEE and McGraw-Hill, New York, 1995

8 N. Karunanithi, D. Whitley, Y. Malaiya, "Using Neural Networks in Reliability Prediction", IEEE Software, vol. 9, no. 4, pp 53-59, July, 1992

9 M. Lu, S. Brocklehurst, B. Littlewood, "Combination of Predictions Obtained from Different Software Reliability Growth Models", in Proceedings of the Tenth Annual Software Reliability Symposium, Phipps Mansion, Denver, CO, June, 1992

10 M. R. Lyu, A. Nikora, "Software Reliability Measurements Through Combination Models: Approaches, Results, and a Case Tool," in Proceedings of the 15th Annual International Computer Software and Applications Conference (COMPSAC91), Tokyo, Japan, September, 1991.

11 M. R. Lyu, A. Nikora, "Applying Reliability Models More Effectively," IEEE Software, vol. 9, no. 4, pp 43-52, July, 1992.

12 J. C. Munson, T. M. Khoshgoftaar, "The Use of Software Complexity Metrics in Software Reliability Modeling", in Proceedings of the 1991 International Symposium on Software Reliability Engineering, Austin, TX, May, 1991

13 J. D. Musa, A. Iannino, K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill, New York, 1987