NASA Contractor Report 181889

SOFTWARE RELIABILITY GROWTH MODELS DOMINATED BY RANDOMNESS

Wenhui Shen and Larry Wilson

OLD DOMINION UNIVERSITY
Norfolk, Virginia

Grant NAG1-750
September 1989

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

Software Reliability Growth Models Dominated by Randomness

Wenhui Shen
Larry Wilson
Department of Computer Science
Old Dominion University
Norfolk, VA 23508-8508

Abstract

The Jelinski-Moranda and Geometric models for software reliability failed the consistency test which we proposed. We challenged these models to take data which comes from a process which they have correctly modeled and to make predictions about the reliability of that process. We found that either model, given data precisely from a process it correctly models, will usually fail to make good predictions. We attribute these problems to randomness in the data used as input to the models and indicate a remedy for this lack of robustness, namely replication of data.

Additional Key Words and Phrases: Growth Models, Software Reliability, Simulation, and Replication

0. Introduction

The Jelinski-Moranda [1] and the Geometric [2] are famous and widely-used models in the field of software reliability. These models assume the software being modeled is a Poisson Process with constant failure rate between two consecutive failures. Both models use the sequence of interfailure times from the debugging process to make maximum likelihood estimates of parameters associated with the models. These estimated parameter values are then used to calculate estimates of reliability measures such as MTTF of the debugged product. That is, they predict the future performance of the software based on the data from the debugging process. The Jelinski-Moranda model is often criticized for requiring an identical failure rate for all bugs but the Geometric model is not subject to this criticism. We will show that even if we assume either model correctly models reliability for a piece of software, we still cannot expect good predictions from that model. Also, we will demonstrate the benefits of replicated debugging as a remedy for this problem. It should also be noted that all software reliability models potentially suffer from the same problem and those that have this problem should benefit from replication.

Both models are intended to be used as prediction systems as described in Abdel-Ghaly, Chan and Littlewood [3]. This paper characterizes a prediction system as consisting of three stages, namely a probabilistic model, a statistical inference procedure for estimating model parameters and a prediction procedure for predicting future interfailure times. All three components are seen as critical to the prediction system. The problem we address is in the second stage of the prediction process.

This research was in part supported by NASA Grant 1-750.

Special thanks are due to G. E. Migneault whose ideas formed the starting point for this work.

1. Maximum Likelihood Equations

The Jelinski-Moranda model assumes that there are N initial bugs in the software, that each bug has the common failure rate of ϕ, and that the failure rate of the program is the number of bugs present multiplied by ϕ. Thus, if i - 1 bugs have been removed, the failure rate is $\lambda_i = (N-i+1) * \phi$. If n errors have been removed then interfailure times $t_1, t_2, \cdots, t_n$ have been generated and these may be inserted into the following likelihood equation which corresponds to this model.

$$L(t_1,t_2,\cdots,t_n;N,\phi)=\prod_{i=1}^{n}[\phi*(N-i+1)*e^{-\phi(N-i+1)t_i}]$$

This likelihood equation may be maximized by letting N = $\hat{N}$ and $\phi = \hat{\phi}$ where $\hat{N}$ and $\hat{\phi}$ form the solution of the following equations. $\hat{N}$ and $\hat{\phi}$ are estimators of N and ϕ.

$$\hat{\phi}=\frac{n}{\hat{N}*\sum_{i=1}^{n}t_i-\sum_{i=1}^{n}(i-1)t_i}$$

$$\sum_{i=1}^{n}\frac{1}{\hat{N}-i+1}=\frac{n*\sum_{i=1}^{n}t_i}{\hat{N}*\sum_{i=1}^{n}t_i-\sum_{i=1}^{n}(i-1)t_i}$$

The Geometric model assumes that the failure rate after removing i-1 bugs is $\lambda_i=\alpha\beta^{i-1}$. Again the data of n interfailure times is inserted into the corresponding likelihood equation.

$$L(t_1,t_2,\cdots,t_n;\alpha,\beta)=\prod_{i=1}^{n}[\alpha*\beta^{i-1}*e^{-\alpha*\beta^{i-1}*t_i}]$$

This likelihood may be maximized by letting $\alpha = \hat{\alpha}$ and $\beta = \hat{\beta}$ where $\hat{\alpha}$ and $\hat{\beta}$ form the solution to the following equations and are used as estimators.

$$\hat{\alpha}=\frac{n}{\sum_{i=1}^{n}\hat{\beta}^{i-1}*t_i}$$

$$\hat{\alpha}=\frac{\sum_{i=1}^{n}(i-1)}{\sum_{i=1}^{n}[(i-1)*\hat{\beta}^{i-1}*t_i]}$$

We show in following sections that neither of these models is robust. That is if you were to debug the same program twice, generating two sequences of interfailure times, then each model may give two very different estimates for its parameters.

2. Simulation of Interfailure Times

Since these models each describe a Poisson Process with constant failure rate λ_i, the probability that the next interfailure time is less than t is $1-e^{-\lambda_i t}$. Thus we may obtain an interfailure time by generating an uniformly distributed random number r between 0 and 1 and solving $r=1-e^{-\lambda_i t_i}$ for t_i [4].

3. Testing the Models

A. Jelinski-Moranda Model tests

We assumed a piece of software which has its reliability correctly modeled by Jelinski-Moranda, with parameters N and ϕ fixed. Thus $\lambda_1=N^*\phi$ and we used the simulation process to generate t_1, decreased λ_1 by ϕ to get λ_2 and simulated to get t_2. After iterating n times we had data representing n interfailure times from one debugging run.

The simulated interfailure times were used as input to the model and $\hat{N}$ and $\hat{\phi}$ were calculated. The predicted values of $\hat{N}$ and $\hat{\phi}$ usually differed greatly from the seeded values and there were large variations among the predictions from different simulations for the same seeded values. Each of the following histograms was constructed by generating 128 sets of interfailure times for each value of n with N fixed at 100 and ϕ fixed at 0.001. The number 128 was chosen arbitrarily and appears to be large enough for our purposes. Each graph is a plot of the probability of $\hat{N}$ versus $\hat{N}$.

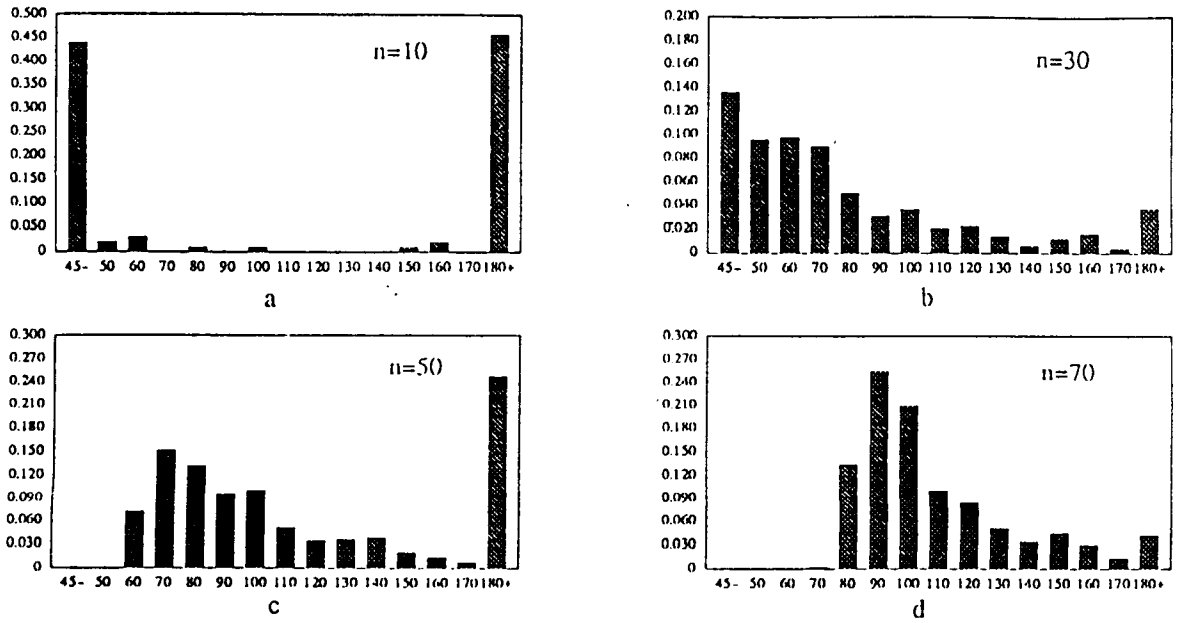
Figure 1.b shows that for N = 100, ϕ = .001 , and n = 30, $\hat{N}$ falls between 95 and 105 less than 5% of the time. For the same graph, $\hat{N}$ falls between 85 and 115 approximately 10% of the time. The other graphs tell a similar discouraging story. As expected, the best estimates are given by the case where n = 70, but even then only about 55% of the estimates are within 15 of 100. We also point out that since 70 errors have been removed, we are actually only trying to estimate the remaining 30; thus our estimates are off by 50% or more 45% of the time.

These results duplicate those of a simulation done by Joe and Reid [5]. We conclude that the model is very sensitive to random variations in the input data even when the data is precisely what the model says it should be. Thus, the Jelinski-Moranda model should not be used to make predictions about software based on a single sequence of interfailure times generated by one debugging run.

B. Geometric Model tests

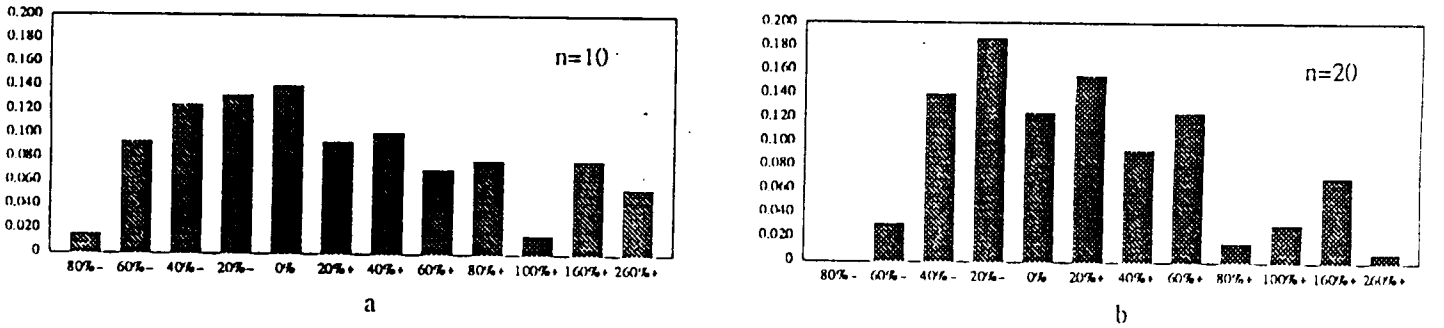
The failure rate for the Geometric model is given by $\lambda_i=\alpha^*\beta^{i-1}$, for each i. We assigned α = 0.1 in order to have the same initial failure rate as that used in the Jelinski-Moranda tests. We chose β = 0.8 as a compromise between the 0.95 which Moranda [2] found for a set of real data and the 0.2 to 0.3 values which appear to be representative of the Nagel [6,7] and Dunham [8,9] data. Data was simulated using the changing λ_i values and the model was used to predict λ_{n+1}, the failure rate of the product after n bugs have been removed. Once again the results of 128 repetitions for each value of n are represented by histograms. In these graphs the y-axis represent the probability of a prediction having a certain percentage of error relative to the correct value of λ_{n+1}. The values of n where chosen to be large enough to illustrate our point but small enough to avoid precision problems in the calculations.

In these histograms the 0% bar represents the proportion of the estimates which fall within plus or minus 10% of λ_{n+1}. For n = 10, only (approximately) 12% of the estimates come within 10% of the desired value. Also for n = 20, only (approximately) 46% of the estimates come within plus or minus 30% of the correct value. This indicates that the Geometric Model also has trouble handling the variations in the data from one debugging to the next.



Jelinski–Moranda Model N=100 Φ=0.001

Fig. 1



Geometric Model α=0.1 β=0.8

Fig. 2

4. Replicated testing

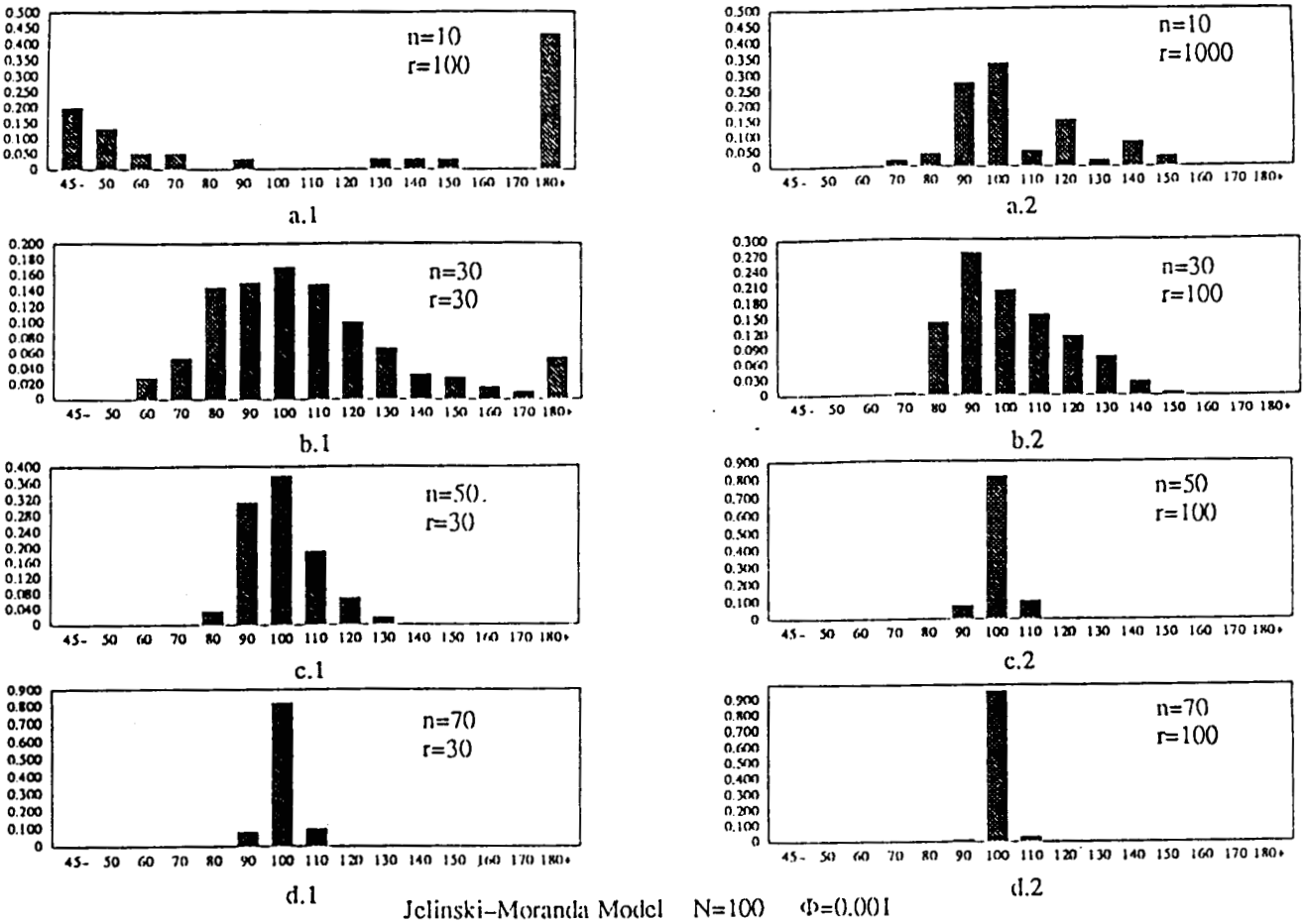
Replicated debugging was introduced in Nagel [6,7] and was used with increased automation in Dunham [8,9]. Rather than attempt to summarize these papers here, we refer the interested reader to the originals. We do note that these papers used multiple programs written to satisfy a common specification in order to produce high quality replicated data which can be used to argue against the uniform failure rates of the Jelinski-Moranda model but appear to support the exponential decline in the failure rates associated with the Geometric model. We believe that replication is itself a very powerful and possibly necessary tool which has not yet been fully appreciated in the field of software reliability.

By replicated debugging we mean the process of repeatedly debugging the same piece of software or more precisely the following. Given a piece of software, make r copies and debug each of them independently (except for shared fixes), removing the bugs from each replicate as they are discovered. For simulation purposes we chose to stop each replicate after generating an interfailure times sequence of length n, thus r replicates generated r sequences of interfailure times of the form $t_{1j}, t_{2j}, t_{3j}, \ldots, t_{nj}$ for 1<= j <= r. Both Nagel and Dunham used random inputs from a known input distribution to generate test cases and counted test cases between failures as the interfailure time. They did not however, remove a fixed number of bugs from each replicate but rather terminated each replicate after a fixed number of test cases or in some cases, for economic reasons, when a rare bug was encountered. Our simulation also represents situations where interfailure times are measured in clock time or in calendar time.

In order to get simulated data representing replicated debugging of a particular model, we assigned values to the necessary parameter and repeatedly simulated sequences of n interfailure times for that model until we had r such sequences. For ease of calculation we reduced this r X n matrix of data to a single sequence of average interfailure times by letting

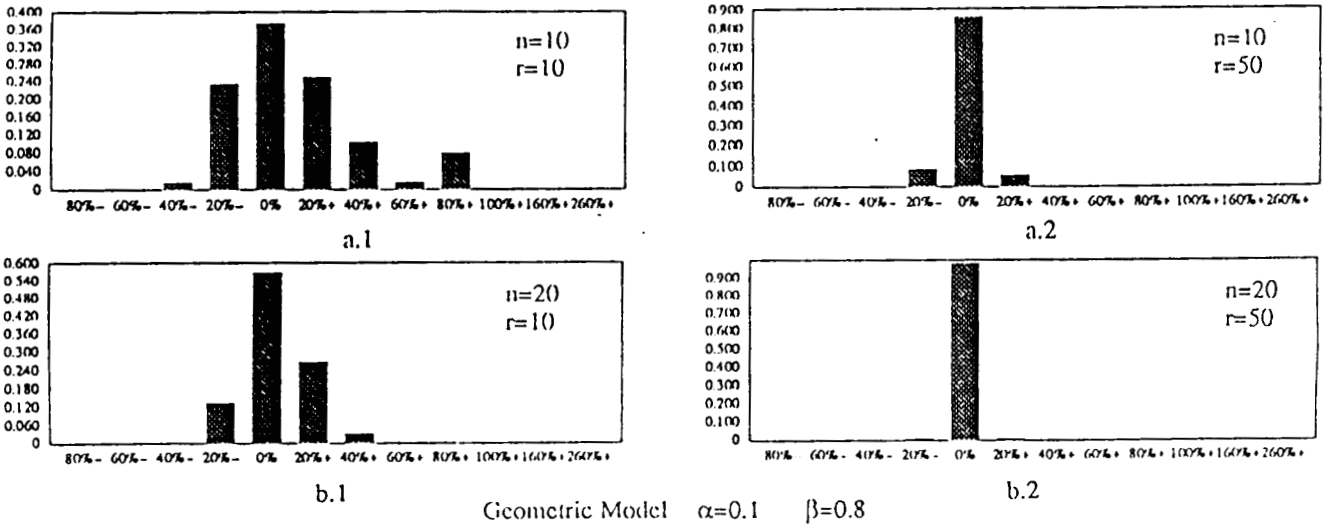
$$t_i = \sum_{j=1}^{r} \frac{(t_{ij})}{r} \quad \text{for} \quad 1 <= i <= n.$$

When we repeated the tests for both models using interfailure data which was the average of r replications instead of from a single debugging, we observed that the models performed monotonically better as r increased. Each of the histograms in figures 3 and 4 was constructed by generating 128 sets of averaged interfailure times. These histograms when considered with those in the previous section indicate that the models require more than the normal debugging data in order to give good predictions and that they also show that replication offers a remedy. In particular, figure 3.d.1 indicates that with 30 replicates the Jelinski-Moranda model with n = 70 gives estimates between 95 and 105 about 80% of the time and almost always gives estimates between 85 and 115.



Jelinski–Moranda Model N=100 Φ=0.001

Fig. 3



Geometric Model α=0.1 β=0.8

Fig. 4

5. Confidence Intervals in Predictions

If we wish to quantify our confidence in the predictions of the models, then we can look at confidence intervals. If we wish to be 90% certain that the estimate is within 10% of the value of the parameter we are trying to predict, then we can achieve this by increasing r or n.

The following graph shows the (n,r) pairs which combine to give estimates within 10% of the value of N-n for the Jelinski-Moranda model with N = 100 and ϕ = .001. It is based on 2,500 repetitions for each (n,r) pair displayed. This graph shows that good predictions are possible for simulated data with replication and it also indicates that without replication one should not expect good predictions.

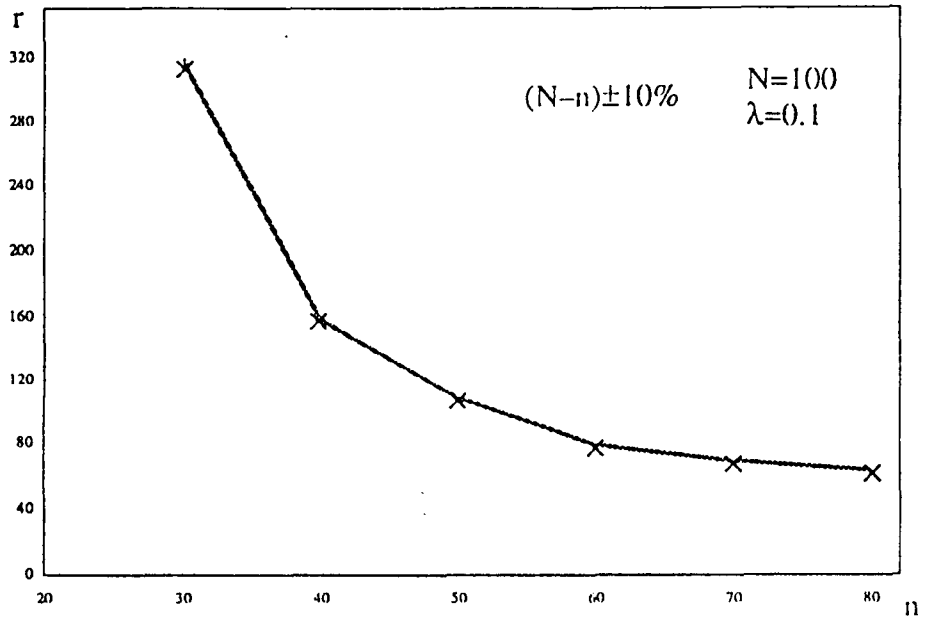


Fig. 5 Confidence Interval Graph

A similar graph could be constructed for the Geometric Model. It too would support the need for replication and show that by increasing either n or r one can obtain better estimates. The Nagel experimental design generates replicated data efficiently by exploiting failure information and fixes discovered during the normal debugging process. This process was automated by Dunham in order to gather replicated data even more efficiently. Further, this could run in the background or in parallel with the debugging effort, and thus it will be less expensive in both time and money to increase r rather than n.

6. Summary

Neither the Jelinski-Moranda model nor the Geometric model should be used to make predictions without replication. This does not guarantee that either model with replication will give good estimates, since the goodness of fit problem has not been addressed here and previous efforts to validate these models have not used replicated data and hence are suspect. It is clear from this work that random chance is likely to dominate if one uses only the data from a single debugging run. It is also claimed that the field of software reliability has been hindered by the random nature of the data and that replication offers a solution to this problem by removing the randomness from the data.

References

1. Z. Jelinski and P. Moranda, "Software Reliability Research," in Statistical Computer Performance Evaluation, W. Freiberger, Ed. New York: Academic, 1972, pp. 465-484.

2. P. B. Moranda, " Prediction of Software Reliability During Debugging," Proceedings of the 1975 Annual Reliability and Maintainability Symposium.

3. Abdalla A. Abdel-Ghaly, P.Y. Chan and Bev Littlewood, "Evaluations of Competing Software Reliability Predictions," IEEE Transactions on Software Engineering, Vol. SE-12, No. 9, September 1986, pp 950-967.

4. Robert V. Hogg and Allen T. Craig, "Introduction to Mathematical Statistics," Macmillan Publishing Co., Inc., New York, 1978, pp126-127.

5. H. Joe and N. Reid, "Estimating the Number of Faults in a System," Journal of the American Statistical Association, Vol. 80, No 389, Theory and Methods, March 1985, pp 222-226.

6. Phyllis M. Nagel and James A. Skrivan, "Software Reliability: Repetitive Run Experimentation and Modeling" NASA Contractor Report 165836, Feb 1982.

7. Phyllis M. Nagel, Fritz W. Scholz and James A. Skrivan, "Software Reliability: Additional Investigations Into Modeling with Replicated Experiments," NASA Contractor Report 172378, June 1984.

8. Janet R Dunham, "Experiments in Software Reliability: Life- Critical Applications," IEEE Transactions on Software Engineering, Vol. SE-12, No 1, January 1986, pp110-123.

9. Janet R Dunham and John L. Pierce, "An Experiment in Software Reliability," NASA Contractor Report 172553, March 1985.

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181889		

4. Title and Subtitle	5. Report Date
Software Reliability Growth Models Dominated by Randomness	September 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Wenhui Shen Larry Wilson	
	10. Work Unit No. 505-66-21-03

9. Performing Organization Name and Address	11. Contract or Grant No.
Old Dominion University Norfolk, VA 23508-8508	NAG1-750
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Technical Monitor: Kelly J. Hayhurst

16. Abstract

The Jelinski-Moranda and Geometric models for software reliability failed the consistency test which we proposed. We challenged these models to take data which comes from a process which they have correctly modeled and to make predictions about the reliability of that process. We found that either model, given data precisely from a process it correctly models, will usually fail to make good predictions. We attribute these problems to randomness in the data used as input to the models and indicate a remedy for this lack of robustness, namely replication of data.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Growth Models Software Reliability Simulation Replication	Unclassified-Unlimited Subject Category - 65

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	9	A02