LANGLEY
GRANT
IN-61-CR
193360
11P.

DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF SCIENCES
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

SOFTWARE RELIABILITY STUDIES

By

Larry W. Wilson, Principal Investigator

Progress Report
For the period ended March 31, 1989

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NAG-1-750
Kelly J. Hayhurst, Technical Monitor
ISD-Systems Validation Methods Branch

March 1989

SOFTWARE RELIABILITY STUDIES

by

Larry W. Wilson

Progress Report
For the period ended March 1989

Prepared for NASA
Langley Research Center
Hampton, Va 23665

Research Grant NAG 1-750
Kelly J. Hayhurst, Technical Monitor
ISD-Systems Validation Methods Branch

The immediate goals of the research being conducted under NASA grant NAG 1-750 are two, namely to create data useful to the study of sotware reliability and to produce results pertinent to software reliability through the analysis of existing reliability models and data. The long term target of this research is to identify and or create a model for use in analyzing the reliability of flight control software.

The data creation portion of this research to date consists of a GCS design document created by Wenhui Shen and work by Larry Wilson in the supervision of Shen and interfacing with the NASA and RTI experimenters. This will shortly lead to design and code reviews with the resulting product being one of the versions used in the Terminal Descent Experiment being conducted by the SVMB of NASA/Langley. Further efforts are being expended parallel to the NASA experiment, with the intent of producing more versions in undergraduate classes at ODU. This parallel work is being conducted by C. M. Overstreet at ODU, aided by Brenda Ellis. Brenda is supported by a Minority Student Research Grant, which is an auxiliary to this grant.

The analysis at this time is being done by Wilson and Wenhui. This has resulted in a recent paper which has been submitted to the ODU CS review process for a technical report. It is expected to be approved any day now and a copy is enclosed, even though it is not yet officially a technical report. The title of the paper is 'Simulation Studies of Software Reliability Models'. This paper has also been submitted to the International Working Conference on Dependable Computing For Critical Applications in Santa Barbara on Aug 23 through 25,1989. Dennis Link has recently been added to the group working on projects related to this grant. Dennis will begin by looking into new ways to exploit the residue from the previous experiment conducted by RTI for SVMB. This experiment involved three versions of the Launch Interceptor Program and the residue is perceived as being very valuable in ways which have not yet been exploited. Further it is expected that the experience we get in looking at this previous experiment will prove useful in preparing us to plan and analyze the data created by the Terminal Descent Experiment.

References

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

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

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

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

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

6. 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.

7. Wilson, Larry W. and Shen, Wenhui, "Software Reliability Perspectives", Old Dominion University Computer Science Department # TR-87-035, 1987.

8. Shen, Wenhui and Wilson, Larry W., "Simulation Studies of Software Reliability Models". submitted to ODU TR series.

Simulation Studies of Software Reliability Models

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 [3] and the Geometric [4] 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 use the sequence of interfailure times from the debugging process to make maximum likelihood estimates of parameters associated with the models. That is they predict the future performance of the software based on the data from the debugging process.

The Jelinski-Moranda model assumes that there are N initial bugs in the software, that each 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 remove 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 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.

The likelihood equation may be maximized by letting N = $\hat{N}$ and ϕ = $\hat{\phi}$ where $\hat{N}$ and $\hat{\phi}$ form the solution of the following equations and are used as 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 α = $\hat{\alpha}$ and β = $\hat{\beta}$ 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 could give very different estimates for its parameters.

1. Simulation of Interfailure Times

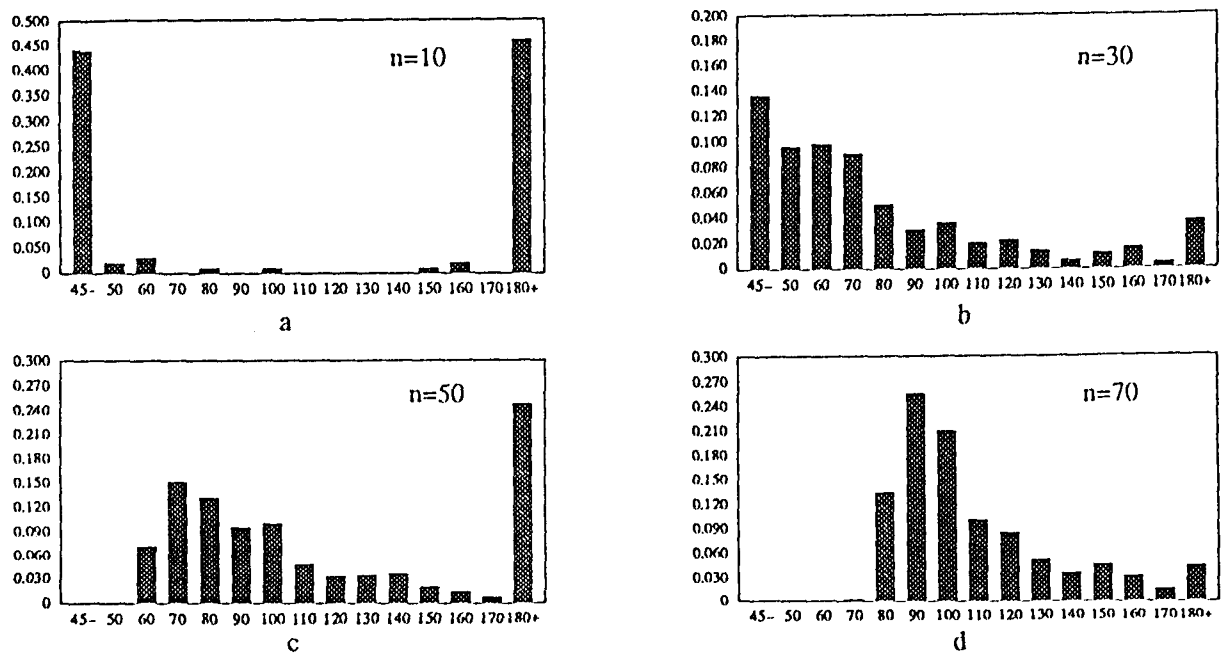
Since these models each describe a Poisson Process with constant failure rate λ_i, the probability that the next interfailure time is less than t is $1-e^{-\lambda_i t}$. Thus we may obtain an interfailure time by generating an uniformly distributed random number r between 0 and 1 and solving $r=1-e^{-\lambda_i t_i}$ for t_i [7].

2. 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 the debugging process.

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 amongst 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. Each x axis represents values of $\hat{N}$.



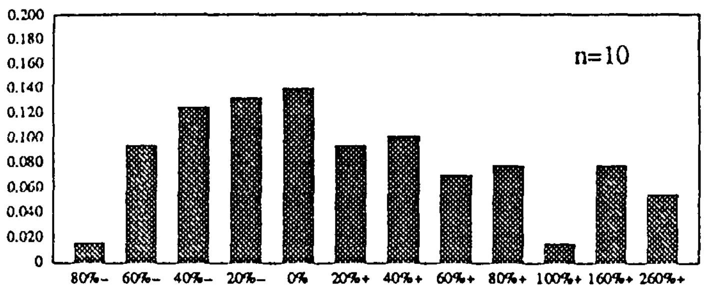
Jelinski–Moranda Model N=100 Φ=0.001

Fig. 1

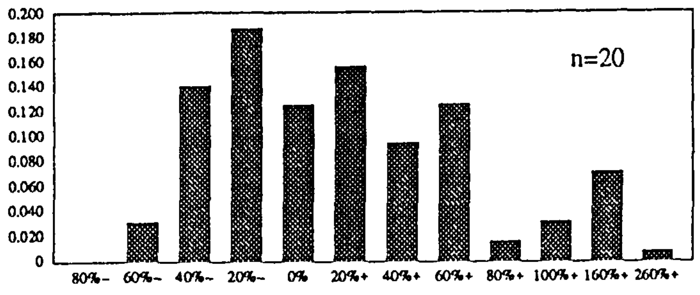
Figure 1.b shows that for N = 100, ϕ = .001 , and n = 30, $\hat{N}$ falls between 95 and 105 less than 5% of the time. Further for the same graph, $\hat{N}$ falls between 85 and 115 approximately 10% of the time. The other graphs tell a similar discouraging story. As expected, the best estimates are given by the case where n = 70, but even then only about 55% of the estimates are within 15 of 100. We must 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. 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 the interfailure times of the debugging process.

B. Geometric Model tests

In this case α and β were assigned arbitrarily but realistic values, $\alpha = 0.1$, $\beta = 0.8$, and for each i, $\lambda_i = \alpha * \beta^{i-1}$. Data was simulated using the changing λ_i values and the model was used to predict λ_{n+1}, the reliability 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 x-axis measures percentage error of the estimated value from the correct value of λ_{n+1}.



a



b

Geometric Model α=0.1 β=0.8

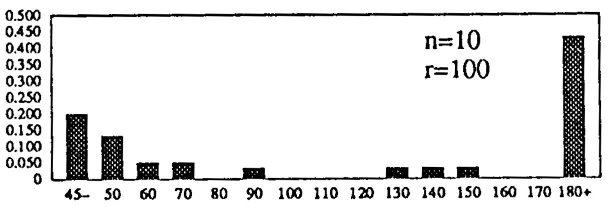
Fig. 2

In these histograms the 0% bar represents the proportion of the estimates which fall within plus or minus 10% of λ_{n+1}. For $n = 20$, 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.

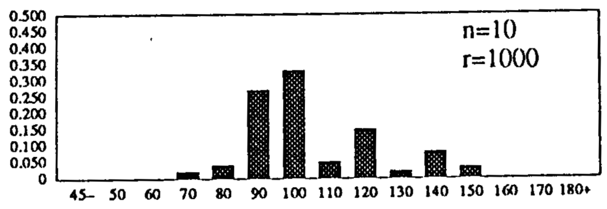
3 Replicated testing

Nagel [5,6] introduced the idea of replicated debugging, which we characterize as follows.. Given a piece of software, make r copies, debug each of the copies for a period of time, which generates r sets of replicated interfailure data. The overhead for approximating this process is less than one might anticipate since all but the normal debugging effort can be automated.

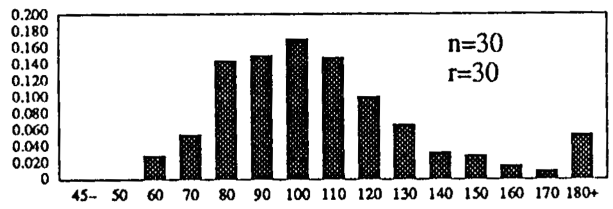
If we repeat the tests for both models using interfailure data which is the average of r replications instead of from a single debugging we now see that the models perform much better as r increases. Each of the histograms in figures 3 and 4 was constructed by generating 128 sets of average interfailure times, each of which was formed by averaging the corresponding interfailure times from r replicates. 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 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. This pattern holds throughout, we get better estimates by either model when we increase r.



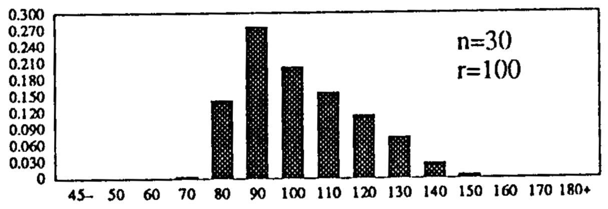
a.1



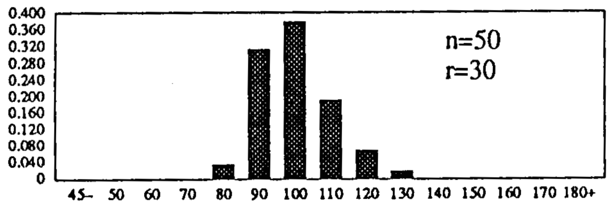
a.2



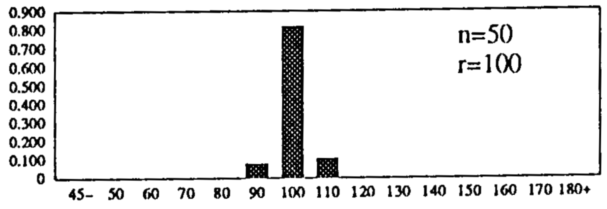
b.1



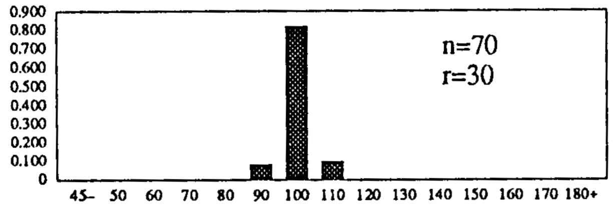
b.2



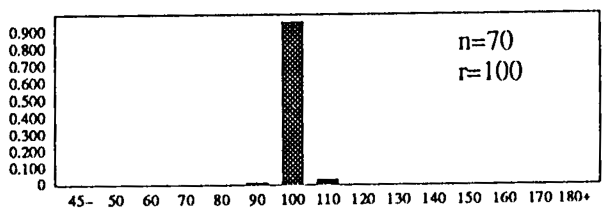
c.1



c.2

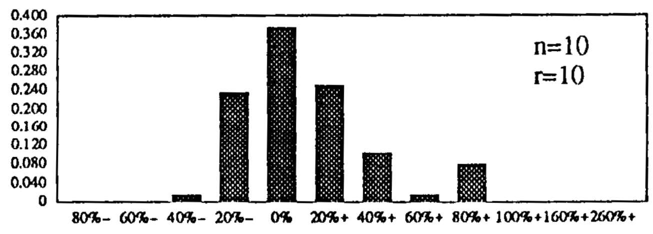


d.1

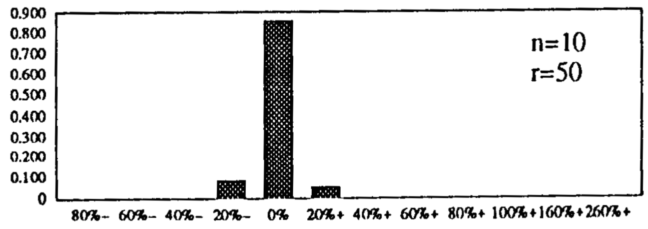


d.2

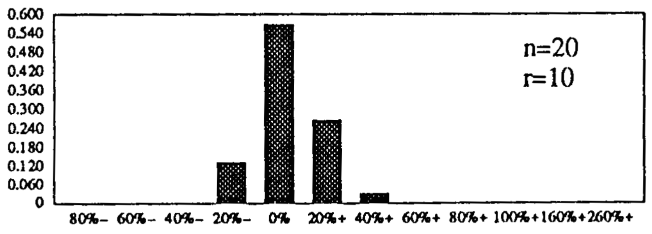
Jelinski–Moranda Model N=100 Φ=0.001
Fig. 3



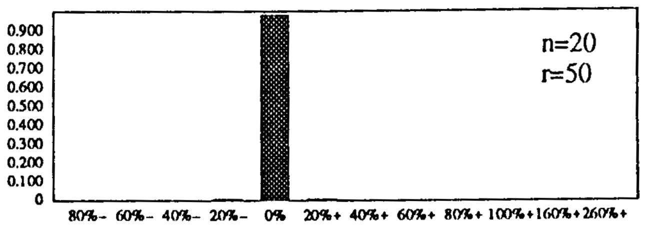
a.1



a.2



b.1



b.2

Geometric Model α=0.1 β=0.8
Fig. 4

4 Confidence Intervals in Predictions

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

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.

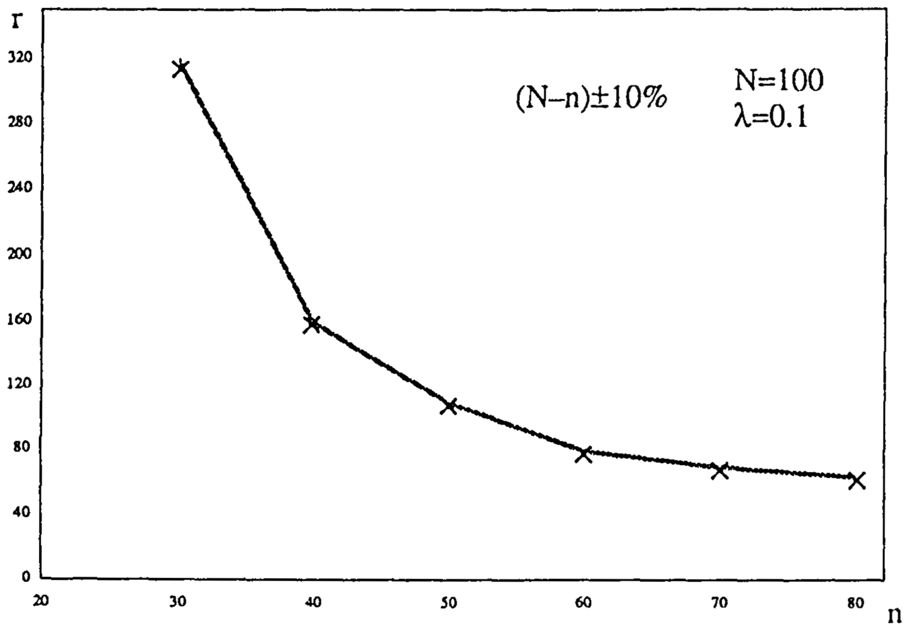


Fig. 5 Confidence Interval Graph

This graph shows that good predictions are possible for simulated data with replication. It also shows that without replication one should not expect good predictions.

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. However, the Nagel experimental design generates replicated data efficiently by exploiting information discovered during the normal debugging process. She uses the failure information and fixes generated during debugging to automate the process of replication. Further, this could run in the background or in parallel with the debugging effort, and thus it should be less expensive in both time and money to increase r rather than n.

5 Summary

These models should not be used in the real world to make predictions without replication. This does not guarantee that the models 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 one replicate. 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. Janet R Dunham, "Experiments in Software Reliability: Life- Critical Applications," IEEE Transactions on Software Reliability, Vol SE-12, No 1, Jan 1986, pp110-123.

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

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

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

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

6. 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.

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