Model Validation Using Simulated Data

Swapna S. Gokhale[1], Michael R. Lyu[2], Kishor S. Trivedi[1]

[1] Dept. of Electrical and Computer Engg.
Center for Adv. Comp. and Commn.
Duke University, Durham
{ssg,kst}@ee.duke.edu

[2] Computer Science & Engineering Dept.
The Chinese University of Hong Kong
Shatin, Hong Kong
lyu@cse.cuhk.edu.hk

Abstract

Effective and accurate reliability modeling requires the collection of comprehensive, homogeneous, and consistent data sets. Failure data required for software reliability modeling is difficult to collect, and even the available data tends to be noisy, distorted and unpredictable. Also, the complexity of the real world data might obscure the properties of the reliability models which are based on simpler assumptions. These properties may be revealed by evaluating the models using simpler data sets. Towards this end, we have created 20 sequences of interfailure times each from five software reliability models using rate-based simulation technique, and validated the models using the simulated data sets. In this paper, we describe the experimental set-up, model validation results, and the lessons learned during the experiment. Having established the credibility of simulation to generate failure data, we also show how the failure process underlying a failure data set can be described more accurately by simulating it using a combination of reliability models, as opposed to a single model as per conventional analytical techniques.

1 Introduction

Effective and accurate reliability modeling requires comprehensive, complete and consistent data sets. A plethora of software reliability models to predict the reliability and error content of software systems have appeared in the literature in the last 20 years, however, an extensive validation of these models seems to be lacking. The accuracy of these models when validated using the very few available data sets varies significantly, and thus despite the existence of numerous models, none of them can be recommended unreservedly to potential users.

Failure data required for software reliability modeling are very difficult to collect because of a number of reasons. Industrial organizations are reluctant to release their reliability data for use by external parties, in the fear that it may reflect badly on the quality of the software product. As the length of the software life-cycle reduces due to the various marketing pressures, the amount of resources expended in testing and validation shrink, further jeopardizing the collection of failure data. Even when data are available, they are often noisy, distorted, and unpredictable. Data collected during the testing and validation phases of the actual projects are influenced by process and product characteristics that are not taken into account in most models. The complexity of real world failure data thus might obscure the properties of software reliability models that might be revealed by validating these models using simpler data sets.

Rate-based simulation offers a viable mechanism to supply carefully controlled, homogeneous data sets, with known characteristics which can be used to evaluate the various assumptions of existing reliability models. We have created sets of interfailure times using simulation from five very popular software reliability models, executed the same five models on each of these sequences, analyzed the behavior of the models on the simulated data, and obtained model validation results. In this paper we discuss the experimental set-up, including the difficulties encountered, the results obtained, and the lessons learned in choosing the most appropriate reliability model. We also compare the results of this controlled experiment, with the validation results obtained using the interfailure data sets from the Handbook of Software Reliability Engineering [3]. After having demonstrated the utility of simulation to generate failure data, we also show how simulation can be used to reproduce the failure behavior underlying a failure data set from a software project more accurately using a combination of reliability models, rather than a single reliability model as in case of the conventional analytical techniques.

The sequel of the paper is organized as follows: Section 2 presents a brief overview of the NHCTMC processes, rate-based simulation for these processes, and the NHCTMC models used in this study, Section 3 describes the experimental set-up, and discusses the results of the experiment, Section 4 presents the model validation results using real data, Section 5 demonstrates the ability of simulation to reproduce the failure process underlying a failure data set collected during the test phase of a software project, and Section 6 presents conclusions and directions for future research.

2 NHCTMC Processes

2.1 Overview

We consider a class of non-homogeneous continuous time Markov chain (NHCTMC) processes, where the behavior of the stochastic process $\{X(t)\}$ of interest depends only on a rate function $\lambda(\mathbf{n}, t)$. The rate function $\lambda(\mathbf{n}, t)$ depends on the state of the system at time t which is denoted by $\mathbf{n}$, where $\mathbf{n}$ in general is a tuple. Let $X(t)$ be the number of "events" occurring in an interval $(0, t)$. "Events" here refers to the number of times the phenomenon of interest occurs (number of failures, for example) and this number denotes the state of the system. Thus $\mathbf{n}$ in this case is a single number, and will be referred to hereafter as n. $\{X(t)\}$ can be viewed as a pure death process if we assume that the maximum number of events that can occur in the time interval of interest is fixed, and the remaining number of events forms the state-space of the NHCTMC. Thus, the system is said to be in state i at time t, if we assume that the maximum number of events that can occur is N, and $N-i$ events have occurred by time t. It can also be viewed as a pure birth process, if the number of occurrences of the event forms the state space of the system. In this case, the system is said to be in state i at time t, if the event has occurred i number of times up to time t. Let $N_0(0, t)$ denote the cumulative number of events in the interval $(0, t)$, and $m_0(0, t)$ denote its expectation, thus $m_0(0, t) = E[N_0(0, t)]$. The notation $m_0(0, t)$ indicates that the process starts at time $t = 0$, and the subscript 0 indicates no events have occurred prior to that time. Pure birth processes can be further classified as "finite events" and "infinite events" processes (if the events of interest are failures, then finite failures and infinite failures), based on the value that $m_0(0, t)$ can assume in the limit. In case of a finite event pure birth process, the expected number of events occurring in an interval of infinite duration is finite (i.e., $\lim_{t\to\infty} m_0(0, t) = a$, where a denotes the expected number of events that can occur in an infinite interval), whereas in case of an infinite event process, the expected number of events occurring in an interval of infinite duration is infinite (i.e., $\lim_{t\to\infty} m_0(0, t) = \infty$). Although these definitions are presented for specific initial conditions

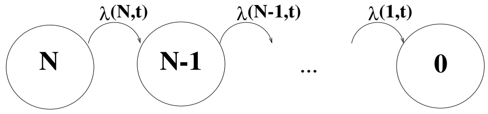

Figure 1. Pure Death NHCTMC

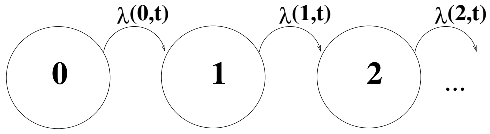

Figure 2. Pure Birth NHCTMC

(the state of the process is 0 at $t = 0$), they hold in the case of more general scenarios. Figure 1 and Figure 2 show the pure death and pure birth NHCTMC, respectively.

2.2 Simulation for NHCTMC Processes

Expressions for the occurrence times of the events of a NHCTMC process are rarely analytically tractable [6]. However, the sample path of a NHCTMC can be easily simulated using rate-based simulation. The occurrence time of a first event of a pure-birth NHCTMC process can be generated using the algorithm shown in the Figure 3, expressed in a C-like form [5]. The function $single\_event()$ returns the occurrence time of the event. In the code segment above, $occurs(x)$ compares a random number with x, and returns 1 if $random() < x$, and 0 otherwise. The algorithm described above can be modified to generate a sequence of interfailure times, or a sample path of the NHCTMC, and the modified algorithm is shown in the Figure 4. In the algorithm shown in Figure 4, simulation is conducted till a certain maximum number of events (denoted by max_events) occur, or up to a certain maximum time (denoted by $tmax$), whichever is earlier. Although the simulation algorithms presented here are for a pure-birth process, they are equally applicable to a pure-death process with suitable modifications.

```
/* Input parameters and functions are assumed to
be defined at this point  */
double single_event(double t, double dt,
                  double (*lambda) (int,double))
{
        int event = 0;
        while (event == 0)
        {
        if (occurs(lambda(0,t) * dt))
              event ++;
        t += dt;
        }
        return t;
}
```

Figure 3. Single Event Simulation Algorithm

```
/* tmax, dt, beta(n,t), and max_events are
assumed to be set at this point */
double *seq_inter_fail(double tmax, double dt,
        double (*lambda) (int,double),
                              int max_events)
{
     int num_events = 0;
     double time_detected[max_events],
     inter_fail_time[max_events], t = 0.0;
     time_detected[0] = 0.0;

     while((num_events < max_events) && (t < tmax))
     {
          if (occurs(lambda(num_events,t)*dt))
          {
               num_events++;
               time_detected[num_events] = t;
          }
          t += dt;
     }
     for(i=0;i<num_events;i++)
        inter_fail_time[i] = time_detected[i+1]
                           - time_detected[i-1];
     return inter_fail_time;
}
```

Figure 4. Simulation Algorithm to Generate Sample Path of a NHCTMC

2.3 NHCTMC Based Software Reliability Models

Some of the popular software reliability models are NHCTMC based, and thus simulation techniques for the analysis of NHCTMC processes offers an attractive approach to study and enhance these models. Table 1 presents a brief overview of the NHCTMC models used in this study, in terms of their rate functions, interpretation of the parameters of the rate functions, and their classification.

3 Experimental Study

In this section, we describe the experimental set-up, and discuss the results of the experiment.

3.1 Experimental Procedure

The experimental set-up consists of the following steps:

1. **Estimating the parameters:** The parameters of the rate functions of the five models described in Section 2.3 are estimated from field data [3]. The field data consisted of 131 failures in 68000 execution minutes.

2. **Generating TBE Sequences:** 20 time between execution (TBE) sequences are generated from each of the five models, resulting in a total of 100 sequences.

3. **Validating the models:** CASRE [4] was used to run the same five models on the sequences. CASRE is a software reliability modeling tool implemented in a Microsoft Windows environment. The core modeling capabilities of this tool are the libraries implemented for version 5 of the software reliability modeling tool SMERFS [2]. Model validation is carried out using both maximum likelihood (ML) as well as least squares (LS) methods of parameter estimation. A recent study by Wood [7] shows that although LS estimates do not possess desirable properties of unbiasedness and minimum variance possessed by the ML estimates, they are numerically more stable than the ML estimates.

4. **Ranking the Models:** The models are then ranked with respect to KS goodness-of-fit [6], bias, bias trend, and prequential likelihood. Model bias, bias trend and prequential likelihood are used to evaluate model applicability [1].

3.2 Results and Discussion

The results of the experiment described in Section 3.1 are summarized in this section. For every data set, an overall rank was computed for each model by equally weighing the ranks according to KS test, bias, bias trend and prequential likelihood (PL). The results of model validation for a set of 20 sequences of interfailure times simulated from a particular model, were summarized by taking averages of the ranks computed for each of the 20 sequences. Model validation results obtained from using maximum likelihood parameter estimation method are summarized in Tables 2-6. Similar results were obtained for validation using the least squares estimation method, but are not presented here, due to space constraints. For Tables 2-6, even-numbered columns give the average rank of the model across all inputs with respect to prequential likelihood, model bias, model bias trend, and KS goodness-of-fit test. Column 1 gives the overall rank of the model, computed by equally weighting the individual ranks according to PL, model bias, model bias trend, and KS test, while odd-numbered columns after the first column give the standard deviation for the model ranking given in the preceding column. Model i is said to be "better" than model j, if the overall rank of model i is higher, or the score in the overall rank column is lower for model i than for model j. The difference in the overall rank of model i and model j will be a judgment of "how better" model i is compared to model j.

JM Model: Table 2 shows that using the data simulated from the JM model and ML estimation method, MB model has the highest overall rank, whereas when the LS method of estimation is used both the MB and the JM model share the highest overall rank. Since the JM model is a finite failures model, models based on the same assumption have a better performance than the infinite failures models.

Table 1. Overview of NHCTMC Models

Model	Rate Function	Interpretation of parameters	Classification
Jelinski-Moranda (JM)	$\lambda(n,t) = \phi(1 - n/n_0)$	n_0 - number of faults ϕ - initial failure rate	Pure death
Musa-Okumoto (MO)	$\lambda(n,t) = \frac{\beta_0}{(1+\theta t)}$	β_0 - initial failure rate θ - rate decay factor	Pure birth infinite failures
Littlewood-Verrall (LV)	$\lambda(n,t) = \frac{\beta_0}{\sqrt{1+\theta t}}$	β_0 - initial failure rate θ - rate decay factor	Pure birth infinite failures
Geometric (Geo)	$\lambda(n,t) = \frac{De^{\beta}}{[D\beta e^{\beta}]t+1}$	D - initial hazard rate $\beta = -ln(\phi)$, ϕ - proportionality constant	Pure birth infinite failures
Musa-Basic (MB)	$\lambda(n,t) = \beta_0\beta_1 e^{-\beta t}$	β_0 - maximum number of faults β_1 - failure occurrence rate per fault	Pure birth finite failures

Table 2. Summary of model rankings - Data Simulated using JM & ML

Model	Overall	SD	PL	SD	Bias	SD	Trend	SD	KS Test	SD
Littlewood-Verrall	2.5500	1.4318	2.5500	1.3563	2.9000	1.1192	3.1500	1.6944	2.5000	1.3179
Musa-Okumoto	3.5500	1.3563	3.5000	1.2773	4.3000	1.3416	2.5000	0.8272	3.1000	1.2524
Geometric	3.1000	1.3338	3.2500	1.4464	3.5500	0.7592	3.1500	1.2680	3.0000	1.6222
Musa-Basic	2.2500	1.2513	2.8500	1.2680	2.0000	1.0761	2.7500	1.6504	3.1500	1.5652
Jelinski-Moranda	2.6000	1.5355	2.8500	1.6631	2.2500	1.4096	3.4500	1.4318	3.2500	1.3328

LV Model: Using the data simulated from LV model and ML estimation method, LV and MO models have the highest overall rank as seen in Table 3, whereas Geo model has the highest overall rank when the LS method is used. LV model is an infinite failures model, and data simulated from this model prefers the models belonging to the infinite failures category.

MO Model: Table 4 shows that using the data simulated from the MO model and ML estimation method, LV has the best overall rank in case of ML method followed by MO model, while MO has the best overall rank when LS method of parameter estimation is used. MO model is an infinite failures model, and hence the data simulated from this model prefers models from the same category.

MB Model: Table 5 shows that using the data simulated from the MB model and ML estimation method, MB model has the highest overall rank, whereas MB and JM have the best overall rank when the LS method is used. MB model belongs to the finite failures category, and hence finite failures models have better performance than the infinite failures models.

Geo Model: Table 6 shows that using the data simulated from the Geo model and ML estimation method, MO model has the best overall rank. MO model also has the best overall rank in case of LS method of estimation. Geo model is an infinite failures model, and hence models belonging to the infinite failures category are preferred.

In conclusion, when the data are simulated from a model which assumes a finite number of failures in a interval of infinite duration, models based on the same assumption are preferred over the models from the infinite failures category. The same holds for models which assume infinite failures in infinite time interval. Across all the 100 data sets, MO model has the highest overall rank followed by LV, Geo, MB and JM models when the ML method is used, while the order is MO, JM, Geo, MB and LV when the LS method is used.

4 Model Validation using Real Data

Model validation was then conducted using data sets from the actual software development projects. We used the data sets compiled on the CD-ROM accompanying the Handbook of Software Reliability Engineering [3]. There are 11 time between execution (TBE) data sets. For most of the TBE data sets, the estimates of the parameters would not converge, which would prevent the computation of other metrics like bias, bias trend etc. One way of overcoming this problem is to move the parameter estimation end point to a value higher than its default. The default data ranges in CASRE use half the data set for estimation, and the remaining half in the prediction phase. For some data sets even after the parameter estimation end point was moved to a value higher than its default, all the models would not converge. Partial results (only for some models) obtained from these data sets were not used to compute the statistics.

When the ML method of parameter estimation is used, the LV model has the highest overall rank followed by the MO, Geo, MB and JM models, i.e., infinite failures models have a better performance over finite failures model. When the LS method is used, the order is MB, MO, JM, Geo and LV. Thus, we can see that a finite failures model has a better performance than a infinite failures model in case of real data and LS method of estimation, whereas the situation

Table 3. Summary of model rankings - Data Simulated using LV & ML

Model	Overall	SD	PL	SD	Bias	SD	Trend	SD	KS Test	SD
Littlewood-Verrall	1.6000	0.8826	1.7000	0.8645	1.9000	1.0712	2.6500	1.5652	2.2500	1.4096
Musa-Okumoto	1.6000	0.3982	1.6500	0.6708	1.5500	0.3104	2.1500	0.9333	2.8500	1.3870
Geometric	2.7000	0.7327	2.6500	0.4894	2.6500	0.3871	3.1500	1.3089	2.6000	0.9947
Musa-Basic	3.8000	0.6156	4.0000	0.0000	3.9000	0.3078	3.3500	1.3870	3.7500	1.3717
Jelinski-Moranda	4.6500	0.7452	5.0000	0.0000	5.0000	0.000	3.7000	1.4179	3.5000	1.5044

Table 4. Summary of model rankings - Data Simulated using MO & ML

Model	Overall	SD	PL	SD	Bias	SD	Trend	SD	KS Test	SD
Littlewood-Verrall	1.5500	0.9445	1.5500	0.8870	1.9000	0.9679	2.4000	1.7290	2.3500	1.0894
Musa-Okumoto	1.9000	0.7881	2.4000	0.3026	2.0000	0.7255	2.6000	0.8208	2.9000	1.4473
Geometric	2.3000	0.8645	2.0500	0.8256	2.1500	0.8751	3.3500	1.0894	3.3000	1.3416
Musa-Basic	3.6500	0.6708	4.0000	0.0000	3.9500	0.2236	3.0000	1.3765	3.0500	1.5035
Jelinski-Moranda	4.7500	0.4443	5.0000	0.0000	5.0000	0.0000	3.6500	1.6311	3.4000	1.5694

is reversed in case of simulated data and LS method. LV model is ranked last in case of both simulated as well as real data, and LS method of estimation.

5 Simulation of Project Data

In this section we demonstrate the ability of simulation to describe the underlying failure process exhibited by data collected from an actual software development project. Simulation allows the flexibility of using a combination of various models till an “adequate fit” is obtained, as opposed to analytical techniques which are restricted to applying a single model to the entire data set.

The algorithm used to choose the most appropriate model or a combination of models from a suite of m models, for a given time interval is described here. Initially, we identify the suite of m NHCTMC models to choose from. We then estimate the parameters of the rate functions of these models from the observed failure data, and then simulate the failure profiles corresponding to each of these m models. The time interval under consideration is then divided into r non-overlapping segments. One extreme situation would be to consider every interfailure time as an interval, in which case, there would as many segments as the number of failures. The other extreme would be to consider all the failures in a single interval, in which case we fit the observed failure profile with only one of the analytical models as per the conventional methods. For every segment, we then compute the sum of square errors (SSE) between the actual and the simulated profiles for each model, and determine the model which provides the lowest value of SSE. This model is then used to simulate the observed failure behavior in that segment. The last step consists of simulating the failure behavior in each of the segments with the model which has the lowest SSE in that segment.

We explain the heuristic developed with the aid of an example. For the sake of simplicity, we assume that the suite consists of two models, viz., Littlewood-Verrall model and the Musa-Okumoto model. The data set consists of 90 failures in an interval of 28126 time units. We divide the entire time interval into two segments, such that the first 45 failures lie in the first segment and the next 45 failures lie in the second segment. The SSE is computed for segments 1 and 2 for both LV and MO models. In case of segment 1, the SSE for LV model is less than the SSE for MO model, whereas in case of segment 2 SSE of MO model is less than SSE of LV model. Thus the LV model is used to simulate the first 45 failures, while the MO model is used to simulate the next 45 failures. If only the LV model is used to fit the entire data, the SSE would be 1304.2, whereas if only the MO model is used to fit the data, the SSE would be 1332.4. If a combination of LV and MO models is used, the SSE is 827.5661. Figure 5 shows the field data, simulated profile of LV model, simulated profile of MO model, and simulated profile obtained using the combination of LV and MO models.

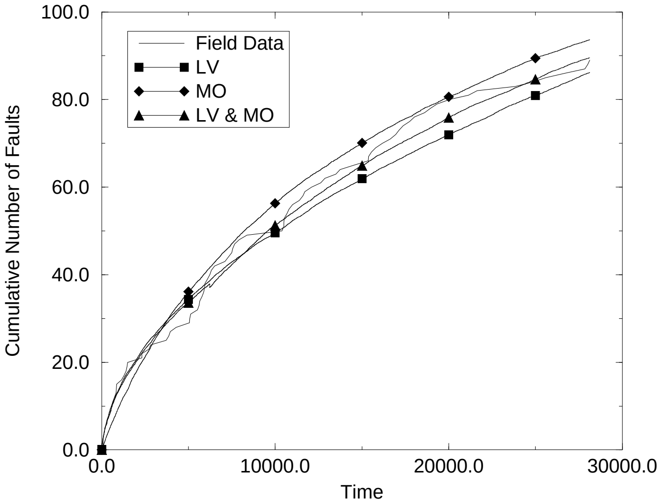

Figure 5. Approximation of Field Data using a Combination of LV and MO models

Table 5. Summary of model rankings - Data Simulated using MB & ML

Model	Overall	SD	PL	SD	Bias	SD	Trend	SD	KS Test	SD
Littlewood-Verrall	2.8500	1.5652	2.4500	1.5035	2.9000	1.1192	3.3000	1.7800	3.4500	1.6051
Musa-Okumoto	3.1500	1.1367	3.4000	1.1425	3.6000	1.5009	2.7500	0.7164	2.9500	1.3945
Geometric	2.7000	1.3416	2.7000	1.1286	3.1000	1.2096	3.2500	1.2513	2.8000	1.1965
Musa-Basic	2.6500	1.3089	3.2500	1.2085	2.6500	1.1367	2.6000	1.7889	2.9500	1.3169
Jelinski-Moranda	3.0000	1.6222	3.2000	1.8806	2.7500	1.9160	3.1000	1.2937	2.8500	1.5985

Table 6. Summary of model rankings - Data Simulated using Geo & ML

Model	Overall	SD	PL	SD	Bias	SD	Trend	SD	KS Test	SD
Littlewood-Verrall	2.3000	1.5594	1.9500	1.1459	2.7500	1.4096	2.6000	1.6670	3.0500	1.6376
Musa-Okumoto	2.0500	0.9445	2.2000	0.6959	2.3000	1.2607	2.4500	1.0501	3.1000	1.2096
Geometric	2.4500	1.0501	2.1500	0.9333	2.4000	0.8826	2.8500	1.2258	3.2500	1.4824
Musa-Basic	3.2000	1.1050	3.9000	0.7182	3.2000	1.3219	3.0500	1.4318	3.0500	1.3563
Jelinski-Moranda	4.3000	1.0809	4.8000	0.6959	4.3500	1.2258	4.0500	1.1910	2.5500	1.4318

6 Conclusions and Future Research

In this paper we have demonstrated the use of simulation as a powerful and simple mechanism for supplying carefully controlled and homogeneous data sets. We have described an experiment where we have generated 100 sequences of interfailure sequences, 20 each from the five software reliability models, validated the models using these 100 sequences as well as real data. Data generated using models which assume finite expected number of failures in infinite time intervals seem to prefer the models based on the same assumption. The same holds for infinite failures models. In addition, we have also developed a heuristic using which the failure process underlying a failure data set can be more accurately described by simulating the profile using a combination of reliability models, rather than a single reliability model as per conventional analytical techniques.

References

[1] A. A. Abdel-Ghally, P. Y. Chan, and B. Littlewood. "Evaluation of Competing Software Reliability Predictions". *IEEE Trans. on Software Engineering*, SE-12(9), September 1989.

[2] W. H. Farr and O. D. Smith. "Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide". Technical Report NSWCDD TR 84-373, Revision 1, Naval Surface Warfare Center Dahlgren Division, September 1993.

[3] M. R. Lyu. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1996.

[4] M. R. Lyu and A. P. Nikora. "CASRE-A Computer-Aided Software Reliability Estimation Tool". In *CASE '92 Proceedings*, pages 264–275, Montreal, Canada, July 1992.

[5] R. C. Tausworthe and M. R. Lyu. *Handbook of Software Reliability Engineering, M. R. Lyu, Editor*, chapter Software Reliability Simulation, pages 661–698. McGraw-Hill, New York, NY, 1996.

[6] K. S. Trivedi. *"Probability and Statistics with Reliability, Queuing and Computer Science Applications"*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[7] A. Wood. "Predicting Software Reliability". *IEEE Computer*, (11):69–77, November 1996.