J. Chosun Natural Sci.
Vol. 7, No. 1 (2014) pp. 67 – 73
http://dx.doi.org/10.13160/ricns.2014.7.1.67

Parameter Estimation and Prediction for NHPP Software Reliability Model and Time Series Regression in Software Failure Data

Kwang-Yoon Song and In-Hong Chang†

Abstract

We consider the mean value function for NHPP software reliability model and time series regression model in software failure data. We estimate parameters for the proposed models from two data sets. The values of SSE and MSE is presented from two data sets. We compare the predicted number of faults with the actual two data sets using the mean value function and regression curve.

Key words: Mean Value Function, MSE, Software Reliability, Regression Curve

1. Introduction

The pioneering attempt in non-homeogenous Poisson process based on software reliability growth models(SRGM) was made by Goel and Okumoto[1]. The model describes the failure observation phenomenon by an exponential curve. There are also SRGM that describe either S-shaped curves, or a mixture of exponential and S-shaped curves (flexible). Goel and Okumoto[1] presented a stochastic model for the software failure phenomenon based on a nonhomogeneous Poisson process (NHPP). Also, they developed a suitable mean-2 value function for the NHPP; expressions are given for several performance measures. Goel-Okumoto compared the NHPP and the Jelinski-Moranda models. Yamada and Osaki[2] summarized existing software reliability growth models (SRGM's) described by nonhomogeneous Poisson processes. And they classified the SGRM's in terms of the software reliability growth index of the error detection rate per error. Yamada and Osaki discussed the maximum-likelihood estimations based on the SRGM's for software reliability data analysis and software reliability evaluation. Pham and Zhang[3] proposed software reliability models based on a nonhomogeneous Poisson process (NHPP) are summarized. They proved that all models are applied to two widely used data sets. It can be shown that for the failure data used here, the new model fits and predicts much better than the existing models. Pham and Zhang[4] presented a cost model with warranty cost, time to remove each error detected in the software system, and risk cost due to software failure is developed. They used a software reliability model based on nonhomogeneous Poisson process. And Pham and Zhang provided the optimal release policies in which the total software system cost is minimized. Pham[5] considered software reliability growth models (SRGMs) incorporating the imperfect debugging and learning phenomenon of developers have been developed by many researchers to estimate software reliability measures such as the number of remaining faults and software reliability. Pham proposed a software reliability model connecting the imperfect debugging and learning phenomenon by a common parameter among the two functions, called the imperfect-debugging fault-detection dependent-parameter model. Grag *et al*.[6] presented a computational methodology based on matrix operations for a computer based solution to the problem of performance analysis of software reliability models (SRMs). They used a set of seven comparison criteria have been formulated to rank various nonhomogenous Poisson process software reliability models proposed during the past 30 years to estimate software reliability measures such as the number of remaining faults, software failure rate, and software reliability. Crag *et al*.

Department of computer science and statistics, Chosun University, Gwangju 501-759, Korea

†Corresponding author : ihchang@chosun.ac.krs
(Received : February 16, 2014, Revised : March 14, 2014, Accepted : March 25, 2014)

selected the optimal SRM for use in a particular case has been an area of interest for researchers in the field of software reliability. Kapur and Pham[7] proposed two general frameworks for deriving several software reliability growth models based on a nonhomogeneous Poisson process (NHPP) in the presence of imperfect debugging and error generation. The proposed models are initially formulated for the case when there is no differentiation between failure observation and fault removal testing processes, and then extended for the case when there is a clear differentiation between failure observation and fault removal testing processes. Kapur and Pham developed a unified framework for two cases, i. e. the case when failure observation or removal are considered as one testing process (GINHPP-1), and case when failure observation and fault removal processes are considered to be two different testing processes (GINHPP-2). Some of the important contributions of these types of models are due to Yamada *et al.*[8], Ohba[9], Kapur and Garg[10], Kapur *et al.*[11], and Pham[12].

Research activities in software reliability engineering have been conducted over the past 30 years, and many statistical models have been developed for the estimation of software reliability[13,14]. Software reliability is a measure of how closely user requirements are met by a software system in actual operation. Most existing models for quantifying software reliability are based purely upon observation of failures during the system test of the software product[1-3,8,9,15-18].

This paper is organized as follows: In Section 2, we propose the mean value functions for NHPP software reliability model and time series regression curve model. In Section 3, the mean square error as goodness-of-fit criteria is presented for model estimation from actual data. We also present parameter estimates and predictive values for the proposed models from two actual data sets. And we compare the predicted number of faults with the actual two data sets using the proposed models. Section 4 presents conclusions in this paper and further research direction in software reliability modeling.

2. Model Description

2.1. Model

In this section, we consider the NHPP software reliability models and time series regression models.

Goel-Okumoto Model:

The Goel-Okumoto model(also called as exponential NHPP model) is based on the following assumptions; 1. All faults in a program are mutually independent from the failure detection point of view. 2. The number of failures detected at any time is proportional to the current number of faults in a program. This means that the probability of the failures for faults actually occurring, i. e., detected, is constant. 3. The isolated faults are removed prior to future test occasions. 4. Each time a software failure occurs, the software error which caused it is immediately removed, and no new errors are introduced.

The mean value function is given by

$$m(t) = a(1-e^{-bt}) \tag{1}$$

where a is the expected total number of faults that exist in the software before testing and b is the failure detection rate or the failure intensity of a fault.

Yamada Imperfect Debugging Model:

The NHPP imperfect debugging model is based on the following assumptions; 1. When detected errors are removed, it is possible to introduce new errors. 2. The probability of finding an error in a program is proportional to the number of remaining errors in the program.

The mean value function is given by

$$m(t) = \frac{ab}{b+\alpha}(e^{\alpha t}-e^{-bt}) \tag{2}$$

where the initial condition $m(0) = 0$, $a(t) = ae^{\alpha t}$ is defined as the error content function of time t during software testing and $b(t) = b$ is a constant error detection rate.

Pham-Zhang Model:

The model[3] assumes that; 1. The error introduction rate is an exponential function of the testing time. In other words, the number of errors increases quicker at the beginning of the testing process than at the end. This reflects the fact that more errors are introduced into the software at the beginning, while at the end testers possess more knowledge and therefore introduce fewer errors into the program. 2. The error detection rate function is non-decreasing with an inflection S-shaped model.

Table 1. Software Reliability Model and Time Series Regression Curve

Type	Model	Mean value function
Software Reliability model	G-O	$m(t) = a(1\text{-}e^{-bt})$
	Yamada Imperfect Debugging	$m(t) = \frac{ab}{\alpha+b}(e^{\alpha t}-e^{-bt})$
	Pham-Zhang	$m(t) = \frac{\left((c+a)[1-e^{-bt}]-\frac{a}{b-\alpha}(e^{-\alpha t}-e^{-bt})\right)}{1+\beta e^{-bt}}$
Regression Curve	Logarithm	$Y_t = a+b\ln t$
	Cubic	$Y_t = a+b_1t+b_2t^2+b_3t^3$
	S	$Y_t = e^{a+\frac{b}{t}}$

Assume the time-dependent fault content function and error detection rate are, respectively, $a(t) = c+\alpha(1-e^{-\alpha t})$, $b(t) = b/a+\beta e^{-bt}$, $m(0) = 0$.

Then the mean value function is given by

$$m(t) = \frac{\left((c+a)[1-e^{-bt}]-\frac{a}{b-\alpha}(e^{-\alpha t}-e^{-bt})\right)}{1+\beta e^{-bt}} \quad (3)$$

Now, we consider the time series regression models. We might use simple linear regression:

$$Y_t = a+bt+w_t$$

where Y_t is total number of failure observed at time t and t is not variable but time, and a and b is parameters. And we consider the logarithm curve model, cubic regression model and s-curve model instead of linear regression model.

The curve models are as follows;

Logarithm: $Y_t = a + b\ln t$,

Cubic: $Y_t = a+b_1t+b_2t^2+b_3t^3$, (4)

S-curve: $Y_t = e^{a+\frac{b}{t}}$

2.2. Criteria for Model Comparisons

In this paper, the analytical expression for the mean value function is derived and the model parameters to be estimated in the mean value function can then be obtained. We are used common criteria for the model estimation of the goodness-of-fit such as the sum of squared errors (SSE), the mean squared error (MSE).

The sum of squared errors and the mean squared error given by

$$SSE = \sum_{i=1}^{n}(m(t_i)-y_i)^2,\ MSE = \frac{\sum_{i=1}^{n}(m(t_i)-y_i)^2}{n-N}$$

where y_i is total number of failure observed at time t_i and according to the actual data and $m(t_i)$ is the estimated cumulative number of failure at t_i for $i = 1,2,...,n$.

The MSE measures the distance of a model estimate from the actual data with the consideration of the number n of observations and the number N of parameters in the model. The lower MSE indicates less fitting error, the lower value of MSE is the better the model fits, relative to other models run on the same data set.

3. Numerical Examples

The Data 1 set, given in Table 2 is the data collected from the test of system T at AT&T. We obtained information for Data 1 in Ehrlich (1993) and Pham (2006). The AT&T's system T is a network-management system developed by AT&T Bell Laboratories that receives data from a telemetry network The network presents telemetry events, such as alarms, facility-performance information, and diagnostic messages, to operators for action. The events are detected by local telemetry interfaces and are transmitted through telemetry network interfaces. Events arrive at System T's central processor as alarm messages. System T expands them by referring to stored information in the database. Using that information, it filters and groups them and then presents them to operators. The operators - alarm, surveillance and control-respond mainly by entering commands to

request more information, to alter the network, or to change data. Database operators update the network description and other data in a system database. Each instance of System T is networked to other instances of System T. To test the system, system testers executed it in a manner that emulated its expected use by Unix system administrators; System T system administrators; database operators; alarm, surveillance, and control operators; field users; and report analysts. Testing was conducted in a controlled-load test environment consisting of a command mix and alarm rate characteristic of field use. Thus, the database for the test environment was based on a snapshot of the database from the beta site (the largest system site(the largest system site), with transaction rates for autonomous events and commands also reflecting use at the beta site. System testers also created operational scenarios, representing software installation, database conversion, maintenance of the transport network, database provisioning, and report generation and analysis, and ran them against background load. The system was not restarted unless necessary so that its stability could be tested under continuous operation.

The Data 2 set, listed in Table 3, was extracted from information about failures in the development of software for the real-time multi-computer complex of the

Table 2. Data 1

t(week)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	3	6	10	14	14	16	16	17	17	19	20	20	21	22

Table 3. Data 2

t(week)	1	2	3	4	5	6	7	8	9	10
data	2	8	14	21	22	23	23	23	26	26
t(week)	11	12	13	14	15	16	17	18	19	20
data	26	27	28	30	30	30	30	30	31	31
t(week)	21	22	23	24	25	26	27	28	29	
data	31	31	31	31	31	31	32	33	34	

Table 4. Parameter Estimates from Data 1

Model	Parameter Estimate	SSE(fit)	MSE
G-O	$\hat{a}$=20.877, $\hat{b}$=0.219	8.40218	1.05027
Yamada Imperfect Debugging	$\hat{a}$=20.877, $\hat{b}$=0.219, $\hat{\alpha}$=0.00001	8.40245	1.20035
Pham-Zhang	$\hat{a}$=78.589, $\hat{b}$=1.060, $\hat{\alpha}$=0.009, $\hat{\beta}$=7.062, $\hat{c}$=8.060	3.11298	0.62260
Logarithm	$\hat{a}$=2.592, $\hat{b}$=7.023	7.08701	0.88588
Cubic	$\hat{a}$=-3.633, $\hat{b_1}$=6.823, $\hat{b_2}$=-0.823, $\hat{b_3}$=0.036	3.95315	0.65886
S	$\hat{a}$=3.073, $\hat{b}$=-2.084	7.56321	0.94540

Table 5. Parameter Estimates from Data 2

Model	Parameter Estimate	SSE(fit)	MSE
G-O	$\hat{a}$=31.238, $\hat{b}$=0.198	51.85387	2.25452
Yamada Imperfect Debugging	$\hat{a}$=29.510, $\hat{b}$=0.218, $\hat{\alpha}$=0.00327	50.84656	2.31121
Pham-Zhang	$\hat{a}$=30.687, $\hat{b}$=0.237, $\hat{\alpha}$=7.510, $\hat{\beta}$=0.000, $\hat{c}$=0.0001	42.53271	2.12664
Logarithm	$\hat{a}$=4.979, $\hat{b}$=8.836	70.02417	3.04453
Cubic	$\hat{a}$=0.589, $\hat{b_1}$=4.941, $\hat{b_2}$=-0.279, $\hat{b_3}$=0.005	70.03075	3.33480
S	$\hat{a}$=3.574, $\hat{b}$=-2.861	29.04012	1.26261

Table 6. Predictive Values in Data 1

T (weeks)	Real Data	G-O	Yamada	Pham-Zhang	Logarithm	Cubic	S
1	3						
2	6						
3	10						
4	14						
5	14						
6	16						
7	16						
8	17						
9	17						
10	19						
11	**20**	**19.000**	**19.001**	**18.920**	**19.433**	**20.200**	**17.880**
12	**20**	**19.369**	**19.371**	**19.528**	**20.044**	**22.518**	**18.165**
13	**21**	**19.666**	**19.668**	**20.130**	**20.606**	**25.806**	**18.409**
14	**22**	**19.904**	**19.906**	**20.726**	**21.126**	**30.282**	**18.621**
SSE (Predict)		7.571	7.553	3.769	1.243	98.080	25.991

Table 7. Predictive Values in Data 2

T (weeks)	Real Data	G-O	Yamada	Pham-Zhang	Logarithm	Cubic	S
1	2						
2	8						
3	14						
4	21						
5	22						
6	23						
7	23						
8	23						
9	26						
10	26						
11	26						
12	27						
13	28						
14	30						
15	30						
16	30						
17	30						
18	31						
19	31						
20	31						
21	31						
22	31						
23	31						
24	31						
25	31						
26	**31**	**31.056**	**31.553**	**30.614**	**33.768**	**33.293**	**31.944**
27	**32**	**31.089**	**31.677**	**30.629**	**34.101**	**34.577**	**32.074**
28	**33**	**31.116**	**31.797**	**30.641**	**34.423**	**36.159**	**32.196**
29	**34**	**31.138**	**31.914**	**30.651**	**34.733**	**38.071**	**32.309**
SSE (Predict)		12.575	6.212	18.807	14.638	38.427	4.401

US Naval Fleet Computer Programming Center of the US Naval Tactical Data Systems (NTDS). We obtained information for this Data 2 in Goel[1] and Pham[12]. The NTDS software consisted of some 38 different modules. Each module was supposed to follow three stages: the production (development) phase, the test phase, and the user phase. The data are based on the trouble reports or 'software anomaly reports' for one of the larger modules, denoted as A-module. The times (days) between software failures and additional information for this module are summarized in Goel[1]. Twenty six soft ware errors were found during production phase and five additional errors during test phase. The 'last' error was found on 4 Jan 1971. Then, one error was observed during the user phase on 1971 20 Sep and two more errors (1971. Oct. 5, 1971 10 Nov) during a subsequent test phase, indicating that a rework of the module had taken place after the user error was found.

We use a MATLAB and SPSS program to perform the analysis and all the calculation for MSE. The param-

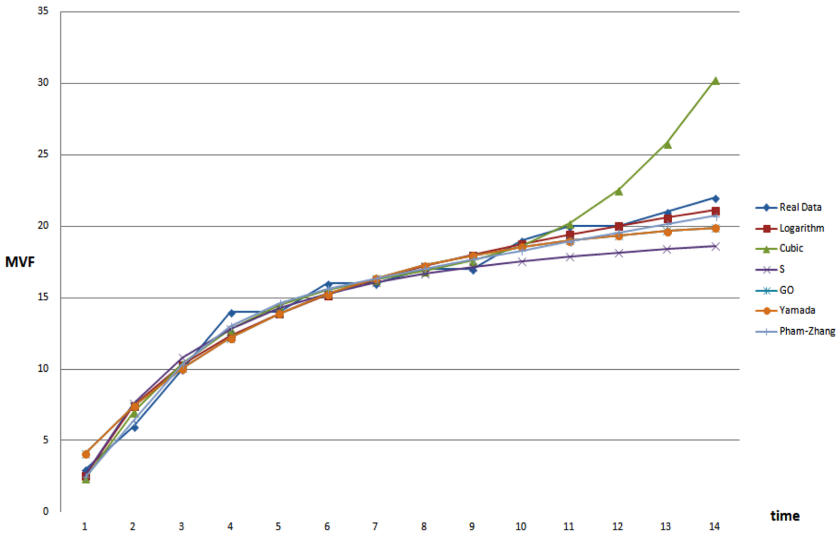

Fig. 1. The Mean Value Function and Time Series Regression Curve from Data 1.

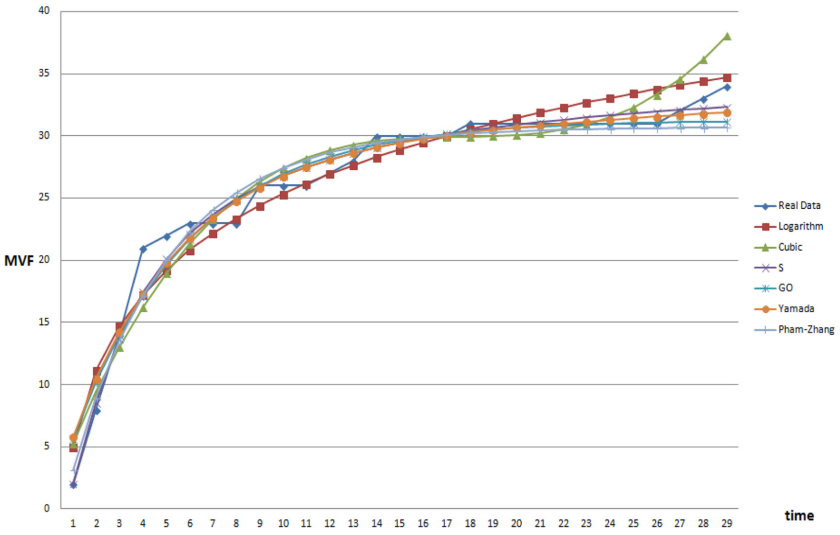

Fig. 2. The Mean Value Function and Time Series Regression Curve from Data 2.

eter estimation results MSE values for goodness-of-fit of the existing models is presented. We obtain MSE when $t = 1,...,t = 10$ from Data 1 (Table 4) and obtain MSE when $t = 1,...,t = 25$ from Data 2 (Table 5).

The 6 models are fitted to the same subset of data to predict the number of future faults: these results are compared. The lower SSE (predict) indicates less prediction error, the lower value of SSE (predict) is the better the model predicts, relative to other models run on the same data set. Table 6 presents the prediction results from week 11 to week 14 in Table 2 (Data 1). The logarithm model predicts better than other model for every week. And Table 7 presents the prediction results from week 26 to week 29 in Table 3 (Data 2). The S model predicts better than other model for every week. Fig. 1 and Fig. 2 shows the graph of mean value functions for 6 models from Data 1 and Data 2, respectively.

4. Conclusions

The mean value functions for NHPP software reliability model and time series regression curve models are considered. We presented parameter estimates for the proposed model. The values of SSE and MSE for models in two data sets is presented. We compared the predicted number of faults the actual two data sets using the mean value functions and time series regression curve. In Data 1, the logarithm model predicts better than other model for every week. The S-model predicts better than other model for every week in Data 2.

Acknowledgement

This study was supported by research funds from Chosun University, 2011.

References

[1] A. L. Goel and K. Okumoto, “Time dependent error detection rate model for software reliability and other performance measures,” IEEE T. Reliab., Vol. R-28, pp. 206-211, 1979.

[2] S. Yamada and S. Osaki, “Software reliability growth modeling: Models and applications”, IEEE T. Software Eng., Vol. 11, pp. 1431-1437, 1985.

[3] H. Pham and X. Zhang, “An NHPP software reliability models and its comparison”, Int. J. Rel. Qual. Saf. Eng., Vol. 4, pp. 269-282, 1997.

[4] H. Pham and X. Zhang, “A software cost model with warranty and risk costs”, IEEE T. Comput., Vol. 48, pp. 71-75. 1999.

[5] H. Pham, “An imperfect-debugging fault-detection dependent-parameter software”, International Journal of Automation and Computing, Vol. 4, pp. 325-328, 2007.

[6] R. P. Grag, K. Sharma, R. Kumar, and R. K. Grag, “Performance analysis of software reliability models using matrix method”, World Academy of Science, Engineering and Technology, Vol. 47, pp. 31-38, 2010.

[7] P. K. Kapur and H. Pham, “A unified approach for developing software reliability growth models in th presence of imperfect debugging and error generation”, IEEE T. Reliab., Vol. 60, pp. 331-340, 2011.

[8] S. Yamada, M. Ohba, and S. Osaki, “S-shaped reliability growth modeling for software fault detection”, IEEE T. Reliab., Vol. 32, pp. 475-484, 1983.

[9] M. Ohba, “Software reliability analysis models”, IBM J. Res. Dev., Vol. 28, pp. 428-443, 1984.

[10] P. K. Kapur and R. B. Garg, “A software reliability growth model for an error removal phenomenon”, Software Engineering Journal, Vol. 7, pp. 291-294, 1992.

[11] P. K. Kapur, R. B. Garg, and S. Kumar, “Contributions to hardware and software reliability”, World Scientific Publishing Co. Ltd., Singapore, 1999.

[12] H. Pham, “System software reliability”, Springer, Berlin, 2006.

[13] A. Wood, “Predicting software reliability”, IEEE Computer, Vol. 11, pp. 69-77, 1996.

[14] H. Pham, “Software reliability”, Springer-Verlag, New Jersey, 2000.

[15] Z. Jelinski and P. B. Moranda, “Software reliability research”, Statistical computer performance evaluation, W. Freiberger Ed., Academic Press, New York, pp. 465-497, 1972.

[16] H. Pham, “Software reliability assessment: imperfect debugging and multiple failure types in software development”, EGandG-RAAM-10737, Idaho National Engineering Laboratory, 1993.

[17] H. Pham, L. Nordmann, and X. Zhang, “A general imperfect software debugging model with S-shaped fault detection rate”, IEEE T. Reliab., Vol. 48, pp. 169-175, 1999.

[18] S. Yamada, K. Tokuno, and S. Osaki, S, “Imperfect debugging models with fault introduction rate for software reliability assessment”, Int. J. Syst. Sci., Vol. 23, pp. 2253-2264, 1992.

[19] W. Ehrlich, B. Prasanna, J. Stampfel, and J. Wu, “Determining the cost of a stop-testing decision”, IEEE Software, Vol, 10, pp. 33-42, 1993.