J. Chosun Natural Sci.
Vol. 7, No. 3 (2014) pp. 193 – 199
http://dx.doi.org/10.13160/ricns.2014.7.3.193

Parameter Estimation and Comparison for SRGMs and ARIMA Model in Software Failure Data

Kwang Yoon Song, In Hong Chang†, and Dong Su Lee

Abstract

As the requirement on the quality of the system has increased, the reliability is very important part in terms of enhance stability and to provide high quality services to customers. Many statistical models have been developed in the past years for the estimation of software reliability. We consider the functions for NHPP software reliability model and time series model in software failure data. We estimate parameters for the proposed models from three data sets. The values of SSE and MSE is presented from three data sets. We compare the predicted number of faults with the actual three data sets using the NHPP software reliability model and time series model.

Key words: ARIMA, Mean Squared Error, Software Reliability, Time Series

1. Introduction

The software is evolving the solution of the immediate problems in a variety of industries and offering the convenience of the customer to satisfy the requirements continue. As the requirement on the quality of the system has increased, the reliability is very important part in terms of enhance stability and to provide high quality services to customers. So the software is plying an ever increasing role in our day today life. Most of the products and services we consume are now based on software or uses software in certain ways[1]. The link between complexity and software faults have been suggested for long, studies as early as 1980s such as Lew *et al*.[2] suggest that software complexity often affects its reliability. Thus while it is important to keep the complexity of software under check, it is also important to tack and monitor their reliability growth. The software testing is still the main source of ensuring reliability and quality of software systems. Software reliability growth models have been used to estimate the reliability change in software products and use the reliability growth predictions for making testing resource allocation decisions. Since the software can rarely be made fully error free, project managers need to balance costs associated with software testing to cost of fixing bugs after release[3]. Software reliability can be modelled using reliability models which can be based on non-homogeneous Poisson process(NHPP). Research activities in software reliability engineering have been conducted over the past 40 years, and many statistical models have been developed for the estimation of software reliability. 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[4-7]. The remarkable achievement in NHPP based on software reliability growth models(SRGMs) was made by Goel and Okumoto[4]. The model describes the failure observation phenomenon by an exponential curve. There are also SRGMs that describe either S-shaped curves, or a mixture of exponential and S-shaped curves (flexible). Goel and Okumoto[4] presented a stochastic model for the software failure phenomenon based on a NHPP. Yamada *et al*.[7] presented two software reliability assessment models with imperfect debugging by assuming that new faults are sometimes introduced when the faults originally latent in a software system are corrected and removed during the testing phase. It is assumed that the fault detection rate is proportional to

Department of Computer Science and Statistics, Chosun University, Gwangju, Korea

†Corresponding author : ihchang@chosun.ac.kr
(Received : August 12, 2014, Revised : September 1, 2014, Accepted : September 25, 2014)

the sum of the numbers of faults remaining originally in the system and faults introduced by imperfect debugging. Pham and Zhang[6] proposed software reliability models based on a 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. In recent, Song and Chang[8] deal with software reliability model and time series regression model.

This paper is organized as follows: In Section 2, we propose the functions for NHPP software reliability model, autoregressive moving average model ARMA (p, q) and autoregressive integrated moving average model ARIMA(p, d, q). 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 tree actual data sets. And we compare the predicted number of faults with the actual three data sets using the proposed models. Section 4 presents conclusions in this paper.

2. Models

In this section, we consider the NHPP software reliability model and time series ARMA and ARIMA models.

2.1. NHPP Software Reliability Model

2.1.1. Goel-Okumoto Model

The Goel-Okumoto model[4] 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.

2.1.2. Imperfect Debugging Model

The NHPP imperfect debugging model[7] 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

$$m(t) = a(1+\alpha t)\left(1-\frac{\alpha}{b}\right)+\alpha a t \tag{2}$$

where the initial condition $m(0)=0$, $a(t)=a(1+\alpha t)$ is defined as the error content function of time t during software testing and $b(t)=b$ is a constant error detection rate.

2.1.3. Pham-Zhang Model

The model[5] 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.

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

Then the mean value function is given by

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

2.2. Time Series Model

Now, we consider the time series models. A time series $\{Z_t,\ t=0,\ \pm1,\ \pm2,...\}$ is autoregressive moving average ARMA(p, q). If it is stationary and

$$Z_t = \varnothing_1 Z_{t-1}+\ldots+\varnothing_p Z_{t-p}+a_t+\theta_1 a_{t-1}+\ldots+\theta_q a_{t-q} \tag{4}$$

with $\varnothing_p \neq 0, \theta_q \neq 0$ and $\sigma_t^2 > 0$. The parameters p and q are called the autoregressive and the moving average orders. a_t, $t=0,\ \pm1,\ \pm2,...$, is a Gaussian white noise sequence. When q=0, the model is called an autoregressive model of order p, AR(p), and when p=0, the model

Table 1. Software reliability model and time series model

Type	Model	Function
Software Reliability	Goel-Okumoto	$m(t)=a(1-e^{-bt})$
	Imperfect Debugging	$m(t)a(1-\alpha t)\left(1-\frac{\alpha}{b}\right)+\alpha at$
	Pham-Zhang	$m(t)=\frac{\left((c+a)[1-e^{-bt}]-\frac{a}{b-\alpha}(e^{-\alpha t}-e^{-bt})\right)}{1+\beta e^{-bt}}$
Time Series	ARIMA(1,1,1)	$Z_t=\delta+Z_{t-1}+\varnothing Z_{t-1}+\varnothing Z_{t-2}+a_t-\theta a_{t-1}$
	ARMA(2,1)	$Z_t=\delta+\varnothing_1 Z_{t-1}-\varnothing_2 Z_{t-2}+a_t-\theta a_{t-1}$
	ARIMA(0,1,0)	$Z_t=\delta+Z_{t-1}+a_t$
	ARIMA(p,d,q)	$Z_t=\varnothing_1 Z_{t-1}+\ldots+\varnothing_p Z_{t-p}+a_t+\theta_1 a_{t-1}+\ldots+\theta_q a_{t-q}$

is called a moving average model of order q, MA(q)[9].

The homogeneous non-stationary model has been referred to as the autoregressive integrated moving average model of order (p, d, q) and is denoted as the ARIMA(p, d, q);

$$\varnothing_p(B)(1-B)^d Z_t=\theta_0+\theta_q(B)_{a_t} \quad (5)$$

When p=0, the ARIMA(p, d, q) model is also called the integrated moving average model of order (d, q) and is denoted as the IMA(d, q) model. When d=0, the original process is stationary and we recall the general stationary ARMA(p, q) process in (4)[10].

3. Numerical Examples and Results

3.1. Software Failure Data

The Data set 1, given in Table 2, was reported by Musa[11] based on failure data from a Real-Time Command and Control System(RTC&CS). There are in total 136 faults reported and the time-between failures in second are listed. The Data set 1 represents the failures observed during system testing for 25 hours of CPU time. The delivered number of object instructions for this system was 21700 and was developed by Bell Laboratories. The Data set 2, given in Table 3, was reported by Stringfellow and Andrews[12] based on failure data come from three releases of a large medical record system, consisting of 188 software components. Each component contains a number of files. Initially, the software consisted of 173 software components. All three releases added functionality to the product. Over the three releases, 15 components were added. Between three and seven new components were added in each release. Many other components were modified in all three releases as a side effect of the added functionality.

Table 2. RTC&CS data set[11] - Data set 1

t(hour)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
data	27	43	54	64	75	83	84	89	92	93	97	104	106	111	116	122	122	127	128	129	131	132	134	135	136

Table 3. Release data set[12]- Data set 2

t(week)	1	2	3	4	5	6	7	8	9	10	11	12	13
data	9	14	21	28	53	56	58	63	70	75	76	76	77

Table 4. OCS data set[13] - Data set 3

t(week)	1	2	3	4	5	6	7	8	9	10	11	12
data	10	12	16	22	28	36	40	43	44	50	51	55

In system test and after release, a defect report is actually a failure. This study uses failure data from the last drop of system test and after release. Table shows cumulative number of failures by week in the last drop of system test for all three release. The cumulative number of failures after release are also shown in the last row. Of all Release data set, we used Release 3 data set. The Data set 3, given in Table 4, was reported by Pham[13]. The On-line Communication System (OCS) project at ABC Software Company was completed in 2000. The project consisted of one unit-manager, one user interface software engineer, and ten software engineers/testers. The overall effort for each of the four phases in the software development process of the project can be described as follows: Analysis(7 weeks), Design(8 weeks), Coding(13 weeks), Testing(12 weeks). The data was collected over a period of 12 weeks during which time the testing started and stopped many times. Errors detection is broken down into sub-categories to help the development and testing team to sort and solve the most critical Modification Requests (MRs) first. These sub-categories are referred to as the severity level depending on the nature of the problem with 1 being the most severe problem, with 2 being the major problem and 3 being a minor problem. OCS data set, maps into week, consists of three types of errors: server1, server2, and server3. The observation time (week) and the number of errors detected per week are presented in Pham(2003). Of all OCS data set, we used OCS 3 data set.

3.2. Estimation and Prediction

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 $m(t_i)$ according to the actual data and is the esti-

Table 5. Parameter estimates and MSE from Data set 1

Model	Parameter estimate	SSE(fit)	MSE(fit)
G-O	$\hat{a}$=128.913, $\hat{b}$=0.1561	618.0973	34.3387
Imperfect Debugging	$\hat{a}$=69.991, $\hat{b}$=0.442, $\hat{\alpha}$=0.050	79.5789	4.6811
Pham-Zhang	$\hat{a}$=179.0, $\hat{b}$=0.558, $\hat{\alpha}$=0.013, $\hat{\beta}$=0.213, $\hat{c}$=206.0	78.7691	5.2513
ARIMA(1,1,1)	$\hat{\delta}$=6.423, $\hat{\varnothing}$=0.894, $\hat{\theta}$=0.359	258.0636	14.3369

Table 6. Parameter estimates and MSE from Data set 2

Model	Parameter estimate	SSE(fit)	MSE(fit)
G-O	$\hat{a}$=214.391, $\hat{b}$=0.0445	244.5456	30.5682
Imperfect Debugging	$\hat{a}$=212.370, $\hat{b}$=0.045, $\hat{\alpha}$=0.000001	244.5511	34.9359
Pham-Zhang	$\hat{a}$=165.0, $\hat{b}$=0.678, $\hat{\alpha}$=0.013, $\hat{\beta}$=8.290, $\hat{c}$=127.0	146.5754	29.3151
ARMA(2,1)	$\hat{\delta}$=42.694, $\hat{\varnothing}_1$=1.926, $\hat{\varnothing}_2$=0.981, $\hat{\theta}$=0.988	418.6297	52.3287

Table 7. Parameter Estimates and MSE from Data set 3

Model	Parameter estimate	SSE(fit)	MSE(fit)
G-O	$\hat{a}$=107.693, $\hat{b}$=0.0622	36.9388	5.2770
Imperfect Debugging	$\hat{a}$=107.971, $\hat{b}$=0.062, $\hat{\alpha}$=0.00	36.9391	6.1565
Pham-Zhang	$\hat{a}$=0.001, $\hat{b}$=0.280, $\hat{\alpha}$=0.0, $\hat{\beta}$=1.925, $\hat{c}$=57.12	33.5889	8.3972
ARIMA(0,1,0)	$\hat{\delta}$=4.091	37.5000	4.1667

Table 8. Predictive values in Data set 1

T (hours)	Real data	G-O	Imperfect debugging	Pham-Zhang	ARIMA (1,1,1)
1	27	18.63	25.68	24.20	27.00
2	43	34.57	43.43	41.54	33.42
3	54	48.20	56.09	53.93	56.56
4	64	59.87	65.48	63.01	65.37
5	75	69.85	72.76	70.02	74.11
6	83	78.38	78.69	75.75	85.19
7	84	85.69	83.76	80.70	91.62
8	89	91.93	88.26	85.18	88.31
9	92	97.28	92.41	89.35	93.90
10	93	101.85	96.32	93.32	96.05
11	97	105.76	100.09	97.16	95.67
12	104	109.11	103.76	100.91	100.78
13	106	111.97	107.37	104.58	109.78
14	111	114.42	110.94	108.18	109.83
15	116	116.51	114.48	111.73	115.73
16	122	118.31	118.01	115.23	121.05
17	122	119.84	121.53	118.68	127.71
18	127	121.15	125.04	122.08	124.73
19	128	122.27	128.55	125.44	131.34
20	129	123.23	132.06	128.75	130.77
21	**131**	**124.05**	**135.56**	**132.03**	**131.21**
22	**132**	**124.76**	**139.06**	**135.25**	**133.88**
23	**134**	**125.36**	**142.56**	**138.44**	**136.94**
24	**135**	**125.87**	**146.06**	**141.59**	**140.36**
25	**136**	**126.31**	**149.56**	**144.69**	**144.10**
SSE (Predict)		**352.705**	**450.161**	**150.335**	**106.562**

Table 9. Predictive values in Data set 2

T (weeks)	Real data	G-O	Imperfect debugging	Pham-Zhang	ARMA (2,1)
1	9	9.33	9.34	4.76	9.00
2	14	18.26	18.28	12.42	10.57
3	21	26.79	26.82	22.85	19.04
4	28	34.96	34.98	34.53	27.81
5	53	42.77	42.79	45.34	35.52
6	56	50.24	50.25	54.08	63.19
7	58	57.38	57.38	60.71	64.15
8	63	64.22	64.20	65.79	64.36
9	70	70.75	70.72	69.88	67.98
10	75	77.00	76.96	73.39	73.59
11	**76**	**82.98**	**82.92**	**76.55**	**76.88**
12	**76**	**88.70**	**88.61**	**79.51**	**75.96**
13	**77**	**94.17**	**94.06**	**82.35**	**74.15**
SSE (Predict)		**505.076**	**497.880**	**41.286**	**8.899**

Table 10. Predictive values in Data set 3

T (weeks)	Real data	G-O	Imperfect debugging	Pham-Zhang	ARIMA (0,1,0)
1	10	6.49	6.49	5.68	10.00
2	12	12.60	12.59	11.66	14.25
3	16	18.33	18.33	17.73	16.25
4	22	23.72	23.71	23.64	20.25
5	28	28.78	28.78	29.18	26.25
6	36	33.54	33.54	34.20	32.25
7	40	38.02	38.02	38.60	40.25
8	43	42.22	42.22	42.36	44.25
9	44	46.17	46.17	45.48	47.25
10	**50**	**49.88**	**49.89**	**48.02**	**48.25**
11	**51**	**53.36**	**53.38**	**50.06**	**52.50**
12	**55**	**56.64**	**56.66**	**51.68**	**56.75**
SSE (Predict)		**8.283**	**8.441**	**15.824**	**8.375**

mated 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.

We obtain the time series model of most suitable for Data set 1, Data set 2, and Data set 3, respectively. We use a Visual C++ and SPSS program to perform the analysis and all the calculation for MSE. The parameter estimation results MSE values for goodness-of-fit of the existing models is presented. We obtain MSE when $t = 1,...,\ t = 20$ from Data set 1(Table 2), obtain MSE when $t = 1,...,\ t = 10$ from Data set 2(Table 3) and obtain MSE when $t = 1,...,\ t = 9$ from Data set 3(Table 4).

The 4 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 8 presents the prediction results from week 21 to week 25 in Table 2(Data set 1). The ARIMA(1,1,1) of time series model predicts better than

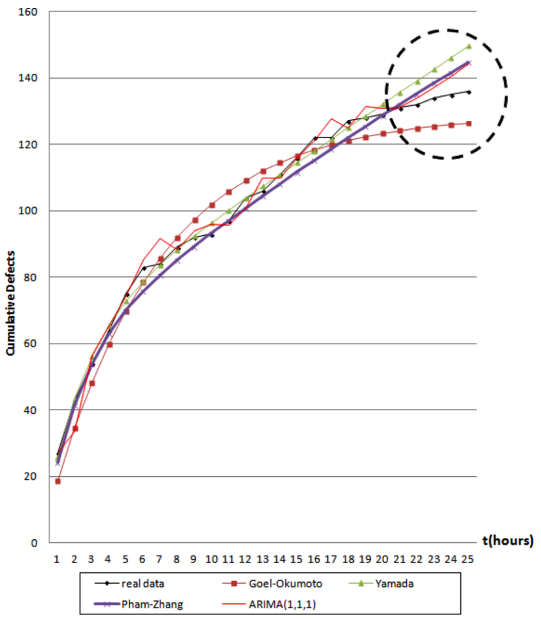

Fig. 1. The mean value function and time series ARIMA (1,1,1) curve from Data set 1.

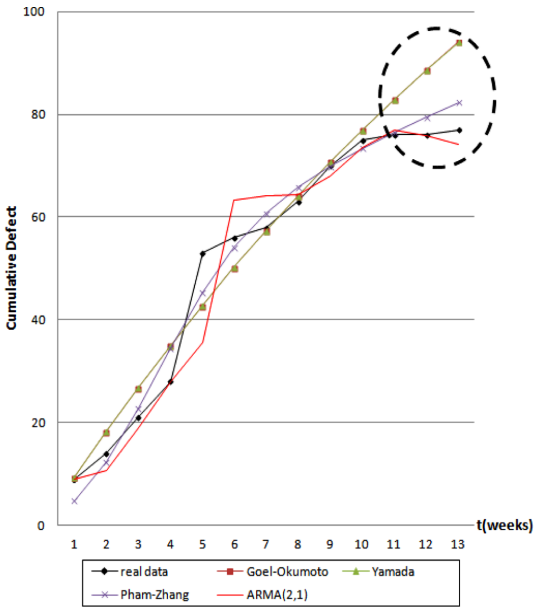

Fig. 2. The mean value function and time series ARMA (2,1) curve from Data set 2.

other model for every week. Table 9 presents the prediction results from week 11 to week 13 in Table 3(Data set 2). The ARMA(2,1) of time series model predicts better than other model for every week, also. Table 10

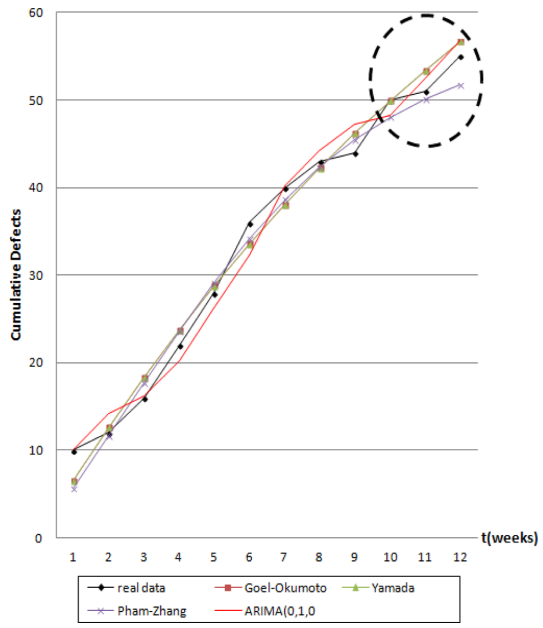

Fig. 3. The mean value function and time series ARIMA (0,1,0) curve from Data set 3.

presents the prediction results from week 10 to week 12 in Table 4(Data set 3). The G-O model predicts better than other model for every week. But the prediction results are very similar values except Pham-Zhang model[5]. Fig. 1, Fig. 2, and Fig. 3 shows the grape(plot) of the failure data from Data set 1, Data set 2, and Data set 3. Also, it shows the curves for the proposed 4 models, respectively.

4. Conclusions

We considered the functions for NHPP software reliability model and time series model. We presented parameter estimates for the proposed models. The values of SSE, MSE for models in three data sets is presented. We compared the predicted number of faults the actual three data sets using the mean value functions and time series ARMA and ARIMA model. In Data set 1, the ARIMA(1,1,1) of time series model predicts better than other model for every week. Also, the ARMA(2,1) of time series model predicts better than other model for every week in Data set 2. In Data set 3, the G-O model predicts better than other model for every week. But we confirmed that the prediction results are very similar values except Pham-Zhang model.

Acknowledgements

This study was supported by research funds from Chosun University, 2013.

References

[1] R. Kitchin and M. Dodge, "Code/space: Software and everyday life", The MIT Press, 2011.
[2] K. S. Lew, T. S. Dillon, and K. E. Forward, "Software complexity and its impact on software reliability", IEEE Trans. Softw. Eng., Vol. 14, pp. 1645-1655, 1988.
[3] T. Goradia, "Dynamic impact analysis: A cost-effective technique to enforce error-propagation", Acm Sigsoft Softw. Eng. Notes, Vol. 18, pp. 171-181, 1993.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] K. Y. Song and I. H. Chang, "Parameter estimation and prediction for NHPP software reliability model and time series regression in software failure data", J. Chosun Natural Sci., Vol. 7, pp. 67-70, 2014.
[9] R. H. Shumway and D. S. Stoffer, "Time series analysis and its applications", Springer, 2006.
[10] W. S. William and Wei, "Time series analysis", Pearson, 2006.
[11] J. D. Musa, A. Iannino, and K. Okumoto, "Software reliability: measurement, prediction, application", McGraw-Hill, New York, 1987.
[12] C. Stringfellow and A. A. Andrews, "An empirical method for selecting software reliability growth models", Empirical Software Engineering, Vol. 7, pp. 319-343, 2002.
[13] H. Pham, "System software reliability", Springer, 2006.