数理解析研究所講究録
第1990巻 2016年 198-203

ロジスティック回帰に基づくテスト環境要因を考慮したソフトウェア信頼性評価に関する一考察

On Software Reliability Assessment Based on Logistic Regression with Testing Environmental Factors

鳥取大学・大学院工学研究科　井上 真二, 山田　茂
Shinji Inoue and Shigeru Yamada
Graduate School of Engineering,
Tottori University

1 Introduction

A software reliability model [11,13] is known as mathematical tool for quantitative assessment of software reliability. In an actual software testing-phase, it must be natural to consider that the software reliability growth process depends on the testing-environmental factors, such as testing-coverage, the number of test-runs, and the debugging skill, which affect the software failure-occurrence or fault-detection phenomenon. In the continuous-time software reliability modeling scheme, a testing-environment dependent software reliability model has been proposed by the literature [4] In the discrete-time domain, Shibata et al. [12] and Okamura et al. [10] proposed extended cumulative Bernoulli trial process models by considering the software metrics in software reliability assessment. On the other hand, the discrete-time models also have been proposed by [12], [10], and [8] only in cumulative Bernoulli trial process modeling approach [2].

In this research, we consider the software complexity which is measured by the number of program size into software reliability assessment. Concretely, we extend the discrete program-size dependent software reliability model following a discrete-time binomial process [5] for incorporating the effect of the testing-environmental factors into the quantitative software reliability assessment and for flexibly depicting a software reliability growth curve described by observed fault counting data. Further, we assume that the discrete software failure-occurrence time distribution basically follows a discrete Weibull distribution for flexibly describing the software failure-occurrence phenomenon, and consider the relationship between the probability that a software failure caused by a fault is observed per the i-th testing-period and the related testing-environmental factors by using a logistic regression modeling approach. This paper also discuss parameter estimation method of our model proposed in this paper. Further, we conduct goodness-of-fit comparisons of our model with the existing corresponding model.

2 Binomial-Type Software Reliability Model

A discrete binomial-type software reliability model [5] is developed based on the following basic assumptions:

(A1) Whenever a software failure is observed, the fault which caused it will be detected immediately, and no new faults are introduced in the fault-detection procedure.

(A2) Each software failure occurs at independently and identically distributed random times I with the discrete probability distribution $P(i) \equiv \Pr\{I \leq i\} = \sum_{k=0}^{i} p_I(k)$ $(i = 0, 1, 2, \cdots)$, where $p_I(k)$ and $\Pr\{A\}$ represent the probability mass function for I and the probability of event A, respectively.

(A3) The initial number of faults in the software system, $N_0(> 0)$, is a random variable, and is finite.

Now, let $\{N(i),\ i = 0,1,\cdots\}$ denote a discrete stochastic process representing the number of faults detected up to the i-th testing-period. From the assumptions above, we have the probability mass function that m faults are detected up to the i-th testing-period as

$$\Pr\{N(i)=m\} = \sum_{n} \binom{n}{m} \{P(i)\}^m \{1-P(i)\}^{n-m} \Pr\{N_0 = n\} \qquad (m = 0,1,2,\cdots,n). \tag{1}$$

In Eq. (1), we consider the case that the probability distribution of the initial fault content, N_0, follows a binomial distribution with parameters (K,λ) which is given as

$$\Pr\{N_0 = n\} = \binom{K}{n} \lambda^n (1-\lambda)^{K-n} \qquad (0<\lambda<1\ ;\ n = 0,1,\cdots,K). \tag{2}$$

Eq. (2) has the following physical assumptions:

(a) The software system consists of K lines of code (LOC) at the beginning of the testing-phase.

(b) Each code has a fault with a constant probability λ.

(c) Each software failure caused by a fault remaining in the software system occurs independently and randomly.

These assumptions are useful to apply a binomial distribution to the probability mass function of the initial fault content in the software system, and to incorporate the effect of the program size into software reliability assessment [7]. The program size is one of the important metrics of software complexity which influences the software reliability growth process in the testing-phase.

Substituting Eq. (2) into Eq. (1), we can derive the probability mass function of the number of faults detected up to the i-th testing-period as

$$\begin{aligned}
\Pr\{N(i)=m\} &= \sum_{n=m}^{K} \binom{n}{m} \{P(i)\}^m \{1-P(i)\}^{n-m} \binom{K}{n} \lambda^n (1-\lambda)^{K-n} \\
&= \binom{K}{m} \{\lambda P(i)\}^m \sum_{n=m}^{K} \binom{K-m}{n-m} \{\lambda(1-P(i))\}^{n-m} (1-\lambda)^{K-n} \\
&= \binom{K}{m} \{\lambda P(i)\}^m \{1-\lambda P(i)\}^{K-m} \qquad (m = 0,1,2,\cdots K).
\end{aligned} \tag{3}$$

From Eq. (3), several types of discrete software reliability model with the effect of program size can be developed by giving suitable probability distributions for the software failure-occurrence times, respectively.

3 Discrete Software Failure-Occurrence Time Distribution with TE

For flexible discrete software reliability growth modeling, we apply a discrete Weibull distribution [9] to the software failure-occurrence time distribution, which is given by

$$P(i) = 1-(1-p_i)^{i^\gamma}, \tag{4}$$

where p_i represents the probability that a software failure caused by a fault is observed per the i-th testing-period, and γ denotes the shape parameter. The discrete Weibull distribution subsumes geometric and Rayleigh distribution as the special cases. In this research, we assume that p_i depends on the testing-environmental factors at the i-th testing-period and the relationship between p_i and the testing-environmental factors can be given by

$$p_i = \frac{1}{1+\exp[-\boldsymbol{\alpha}\boldsymbol{\beta}_i^{\mathrm{T}}]}. \tag{5}$$

In Eq. (5), $\boldsymbol{\beta_i} = (1, \beta_{1,i}, \beta_{2,i}, \cdots, \beta_{n,i})$ represents the n kinds of testing-environmental factors at the i-th testing-period, $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_n)$ is the coefficient vector, and A^{T} the transposed matrix of the matrix A.

4 Software Reliability Assessment Measures

We can derive software reliability assessment measures under the basic assumptions on the software failure-occurrence phenomenon in Eq. (1). The expectation of the number of detected faults, $\mathrm{E}[N(i)]$, is derived as

$$\begin{aligned}\mathrm{E}[N(i)] &= \sum_{z=0}^{n} z \sum_{n} \binom{n}{z} \{P(i)\}^z \{1-P(i)\}^{n-z} \cdot \Pr\{N_0 = n\} \\ &= \mathrm{E}[N_0] P(i). \end{aligned} \tag{6}$$

And its variance, $\mathrm{Var}[N(i)]$, is also derived as

$$\begin{aligned}\mathrm{Var}[N(i)] &= \mathrm{E}[N(i)^2] - (\mathrm{E}[N(i)])^2 \\ &= \mathrm{Var}[N_0]\{P(i)\}^2 + \mathrm{E}[N_0] P(i)\{1-P(i)\}. \end{aligned} \tag{7}$$

A discrete software reliability function is defined as the probability that a software failure does not occur in the time-interval $(i, i+h](i, h = 0, 1, 2, \cdots)$ given that the testing or the operation has been going up to the i-th testing-period. Then, the discrete software reliability function, $R(i, h)$, under the basic assumption in Eq. (1) is derived as

$$\begin{aligned}R(i,h) &= \sum_{k} \Pr\{N(i+h) = k \mid N(i) = k\} \Pr\{N(i) = k\} \\ &= \sum_{k} \left[\{P(i)\}^k \{1-P(i+h)\}^{-k} \sum_{n} \binom{n}{k} \{1-P(i+h)\}^n \cdot \Pr\{N_0 = n\} \right]. \end{aligned} \tag{8}$$

Concretely, we can derive the discrete software reliability function in the case that the initial fault content follows the binomial distribution in Eq. (2) as

$$\begin{aligned}R(i,h) &= \sum_{z=0}^{K} \Pr\{N(i+h) = k \mid N(i) = k\} \binom{K}{z} \{\lambda P(i)\}^z \{1-\lambda P(i)\}^{K-z} \\ &= \sum_{z=0}^{K} \left[\{P(i)\}^z \{1-P(i+h)\}^{-z} \cdot \sum_{n=0}^{K} \binom{n}{z} \{1-P(i+h)\}^n \binom{K}{n} \lambda^n (1-\lambda)^{K-n} \right] \\ &= [1 - \lambda\{P(i+h) - P(i)\}]^K. \end{aligned} \tag{9}$$

Further, instantaneous and cumulative MTBFs, $MTBF_I(i)$ and $MTBF_C(i)$, are also derived as

$$MTBF_I(i) = 1 \Big/ (\mathrm{E}[N(i+1)] - \mathrm{E}[N(i)]), \tag{10}$$

$$MTBF_C(i) = i \Big/ \mathrm{E}[N(i)], \tag{11}$$

respectively.

5 Parameter Estimation Method

Suppose that we have observed N data pairs $(t_i, y_i, \boldsymbol{\beta}_i)(i = 0, 1, 2, \cdots, N)$ with respect to the cumulative number of faults, y_i, detected during a constant time-interval $(0, t_i](0 < t_1 < t_2 < \cdots < t_N)$, and the

related data for the testing-environmental factors, β_i. The likelihood function, l, for the binomial-type software reliability model, $N(i)$, can be derived as

$$\begin{aligned} l &= \Pr\{N(t_1) = y_1, N(t_2) = y_2, \cdots, N(t_N) = y_N\} \\ &= \prod_{i=1}^{N} \Pr\{N(t_i) = y_i \mid N(t_{i-1}) = y_{i-1}\} \Pr\{N(t_1) = y_1\}, \end{aligned} \tag{12}$$

by using the Bayes' formula and a Markov property. In Eq. (12), $t_0 = 0$ and $y_0 = 0$. Thus, $\Pr\{N(t_0) = y_0\} = \Pr\{N(0) = 0\} = 1$. The conditional probability in Eq. (12), $\Pr\{N(t_i) = y_i \mid N(t_{i-1}) = y_{i-1}\}$, can be derived as

$$\Pr\{N(t_i) = y_i \mid N(t_{i-1}) = y_{i-1}\} = \binom{K - y_{i-1}}{y_i - y_{i-1}} \{z(t_{i-1}, t_i)\}^{y_i - y_{i-1}} \{1 - z(t_{i-1}, t_i)\}^{K - y_i}, \tag{13}$$

where

$$z(t_{i-1}, t_i) = \frac{\lambda\{P(t_i) - P(t_{i-1})\}}{1 - \lambda P(t_{i-1})}. \tag{14}$$

Then, we can rewrite Eq. (12) as

$$l = \prod_{i=1}^{N} \binom{K - y_{i-1}}{y_i - y_{i-1}} \{z(t_{i-1}, t_i)\}^{y_i - y_{i-1}} \{1 - z(t_{i-1}, t_i)\}^{K - y_i}, \tag{15}$$

by using Eq. (13). Accordingly, the logarithmic likelihood function can be derived as

$$\begin{aligned} \log l \equiv L &= \log K! - \log\{(K - y_N)!\} \\ &\quad - \sum_{i=1}^{N} \log\{(y_i - y_{i-1})!\} + y_N \log \lambda + \sum_{i=1}^{N} (y_i - y_{i-1}) \log\{P(t_i) - P(t_{i-1})\} \\ &\quad + (K - y_N) \log\{1 - \lambda P(t_i)\}. \end{aligned} \tag{16}$$

When we apply Eqs. (4) and (5) as the discrete software failure-occurrence times distribution, the logarithmic likelihood function can be given as

$$\begin{aligned} L &= \log K! - \log\{(K - y_N)!\} + y_N \log \lambda - \sum_{i=1}^{N} \{(y_i - y_{i-1})!\} \\ &\quad + \sum_{i=1}^{N} (y_i - y_{i-1}) \log\{(1 - p_i)^{t_{i-1}^{\gamma}} - (1 - p_i)^{t_i^{\gamma}}\} + (K - y_N) \log\left[1 - \lambda\{1 - (1 - p_i)^{t_N^{\gamma}}\}\right], \end{aligned} \tag{17}$$

by using Eq. (16). We have to estimate the parameters λ, γ, and $\boldsymbol{\alpha}$ if we can know the program size K. Accordingly, we can obtain the maximum-likelihood estimates $\widehat{\lambda}$, $\widehat{\gamma}$, and $\widehat{\boldsymbol{\alpha}}$ of the unknown parameters λ, γ, and $\boldsymbol{\alpha}$, respectively, by solving the simultaneous likelihood functions with λ, γ, and $\boldsymbol{\alpha}$ numerically.

6 Model Comparisons

We compare the performance of our model for software reliability assessment with the existing corresponding model, which does not consider the effect of the testing-environmental factors, by using two data sets collected from actual software testing-phases. The data sets are respectively called DS1 and DS2. The details of the data are shown as follows:

DS1 : $(t_i, y_i, c_i)(i = 1, 2, \cdots, 22\ ;\ t_{22} = 22, y_{22} = 212, c_{22} = 0.9198)$ where t_i is measublack on the basis of weeks and the program size $K = 1.630 \times 10^5$ (LOC) [3],

Table 1 : Results of model comparisons based on the MSE and AIC.

		MSE	AIC	MLL
DS1	Our Model	27.308	4488.62	-2240.31
	Existing Model	28.256	4509.17	-2251.59
DS2	Our Model	33.954	6112.26	-3052.13
	Existing Model	39.713	6115.94	-3054.97

DS2 : $(t_i, y_i, c_i)(i = 1, 2, \cdots, 24\ ;\ t_{24} = 24, y_{24} = 296, c_{24} = 0.9095)$ where t_i is measublack on the basis of weeks and the program size $K = 1.972 \times 10^5$ (LOC) [3],

where y_i represents the number of faults detected up to t_i and c_i is the C0 testing-coverage attained up to t_i. In this model comparisons we treat the C0 testing-coverage as the testing-environmental factors affecting the software failure-occurrence or fault-detection phenomenon. Thus, we treat that $\beta_i \equiv c_i$. Regarding the actual data, DS1 shows the exponential software reliability growth curve and DS2 shows the S-shaped one. And the existing corresponding model assumes that the software failure-occurrence time distribution follows $P(i) = 1 - (1-p)^{i^\gamma}(i = 0, 1, 2, \cdots)$ in Eq. (3) [6], where p represents the probability that a software failure caused by a fault is observed per one testing-period and γ is the shape parameter of the discrete Weibull distribution.

For quantitative comparisons in terms of fitting performance to the actual data, we use mean square error (abbreviated as MSE) [11] and Akaike information criterion (AIC) [1]. Table 1 shows the results of model comparisons based on the MSE, AIC, and MLL represents the maximum log-likelihood, respectively. From Table 1, we can say our model fits well to the actual data even though the actual data shows the exponential or S-shaped software reliability growth curve.

7 Conclusion

We proposed an extended binomial-type software reliability model with the effect of the testing-environmental factors on the software reliability growth process. Especially, the discrete software failure-occurrence time distribution follows the discrete Weibull distribution basically. Further, we discussed a parameter estimation method of our model, and conducted comparisons of the performance of our model with that of existing corresponding model in terms of MSE. In future studies, we need to check the performance of our model with existing models [8, 10, 12] by using a lot of software fault-counting data with software metrics in the future studies because we have an enough time to obtain the appropriate data sets and conducting numerical experiments.

Acknowledgement

This research was supported in part by the Grant-in-Aid for Scientific Research (C), Grant No. 22510150, from the Ministry of Education, Culture, Sports, Science and Technology of Japan and the Telecommunications Advancement Foundation.

References

[1] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, Vol. AC-19, pp. 716–723, 1974.

[2] T. Dohi, K. Yasui and S. Osaki, "Software reliability assessment models based on cumulative Bernoulli trial processes," *Mathematical and Computer Modelling*, Vol. 38, pp. 1177–1184, 2003.

[3] T. Fujiwara and S. Yamada, "A new testing-path coverage measure — Testing-domain metrics based on a software reliability growth model —," *Proc. 13th IEEE International Symposium on Software Reliability Engineering* (ISSRE'02), pp. 71–75, 2002.

[4] T. Imanaka and T. Dohi, "Burr XII distribution-based software reliability modeling," *Proceedings of the 6th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling* (APARM 2014), pp. 176–183, McGraw-Hill, Taiwan, 2014.

[5] S. Inoue and S. Yamada, "Generalized discrete software reliability modeling with effect of program size," *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, Vol. 37, No. 2, pp. 170–179, 2007.

[6] S. Inoue and S. Yamada, "Discrete program-size dependent software reliability assessment: Modeling, estimation, and goodness-of-fit comparisons," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E90-A, No. 12, pp. 2891–2902, 2007.

[7] M. Kimura, S. Yamada, H. Tanaka and S. Osaki "Software reliability measurement with prior-information on initial fault content," *Transactions of Information Processing Society of Japan*, Vol. 34, No. 7, pp. 1601–1609, 1993.

[8] D. Kuwa and T. Dohi "Generalized logit-based software reliability modeling with metrics data," *Proceedings of the 37th Annual International Computer Software and Applications Conference* (COMPSAC 2013), pp. 246–255, IEEE CPS, 2013.

[9] T. Nakagawa and S. Osaki "The discrete Weibull distribution," *IEEE Transactions on Reliability*, Vol. R-24, No. 5, pp. 300–301, 1975.

[10] H. Okamura, Y. Etani and T. Dohi "A multi-factor software reliability model based on logistic regression," *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering* (ISSRE'10), pp. 31–40, IEEE CPS, 2010.

[11] H. Pham, "Software Reliability,"Springer-Verlag, Singapore, 2000.

[12] K. Shibata, K. Rinsaka and T. Dohi, "Metrics-based software reliability models using non-homogeneous Poisson processes," *Proceedings of The 17th International Symposium on Software Reliability Engineering* (ISSRE'06), pp. 52–61, IEEE CPS, 2006.

[13] S. Yamada, "Software Reliability Modeling — Fundamentals and Applications —,"Springer-Verlag, Tokyo, 2013.