数理解析研究所講究録
第2044巻 2017年 46-51

回帰構造に基づくテスト環境を考慮した二項ソフトウェア信頼性モデルの拡張
Extended Binomial-Type Models with Test Environment Based on Regression Structure

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

In an actual software testing phase, it must be natural to consider that the software reliability growth process depends on the test environment factors, such as testing coverage, the number of test-runs, and debugging skills, which affect the software failure occurrence or fault detection phenomenon. In the discrete-time domain, Shibata et al. [11] proposed a proportional hazard rate modeling approach for extending the cumulative Bernoulli trial process model [2]. Okamura et al. [9] also proposed an extended model by using a logistic regression approach for incorporating the effect of software metrics into p_i, which represents the probability that a fault is detected by the i-th test case and the parameter in the cumulative Bernoulli trial process model. Furthermore, Kuwa and Dohi [7] proposed seven types of extended models by applying generalized logic regression to formulate p_i with software metrics in the cumulative Bernoulli trial process model.

We extend the discrete program size-dependent software reliability model following a discrete-time binomial process [4, 5]. Our extended model enables us to incorporate not only the effect of test environment factors but also the effect of software complexity measures into quantitative software reliability assessment and to flexibly depict a software reliability growth curve described by fault counting data observed. More specifically, we assume that discrete software failure occurrence time distribution basically follows a discrete Weibull distribution for flexibly describing the software failure occurrence phenomenon. We also 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 test environment factors by using a generalized linear modeling approach. Furthermore, we conduct goodness-of-fit comparisons of our models with the existing corresponding model and show application examples of our model for software reliability analysis using actual data.

2 Binomial-Type Software Reliability Model

A discrete binomial-type software reliability model [4] 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 [6]. 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 Extension of Our Model

We need to give a suitable probability mass function in Eq. (3), $P(i)$, that represents the software failure occurrence time distribution in order to develop a specific model from based on our modeling framework in Eq. (3). For flexible discrete software reliability growth modeling, we apply a discrete Weibull distribution [8] to the software failure occurrence time distribution. The probability distribution function of the discrete Weibull distribution is given as

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

where $p_i(0<p_i<1)$ 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 probability p_i depends on the testing period due to the maturity of testing skill, changing fault target, existence of fault prone modules, and so forth.

In this research, we assume that p_i depends on the test environment factors at the i-th testing period. And we consider the following three types of functions for p_i: logistic, probit and complementary log-log functions. In the logistic regression approach, the relationship between p_i and the test environment 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 test environment 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 matrix A. In the probit regression approach, the relationship is given by

$$p_i = \Phi(\boldsymbol{\alpha}\boldsymbol{\beta}_i^{\mathrm{T}}), \tag{6}$$

where $\Phi(\cdot)$ denotes the standard normal distribution. And, $\Phi^{-1}(p_i) = \boldsymbol{\alpha}\boldsymbol{\beta}_i^{\mathrm{T}}$. Furthermore, the complementary log-log regression approach, p_i is given by

$$p_i = 1 - \exp[-\exp\{\boldsymbol{\alpha}\boldsymbol{\beta}_i^{\mathrm{T}}\}], \tag{7}$$

where there exists the following relationship: $\log\{-\log(1-p_i)\} = \boldsymbol{\alpha}\boldsymbol{\beta}_i^{\mathrm{T}}$. It is worth mentioning that the behaviors of the logistic, probit and complementary log-log functions are mostly same around $p_i = 0.5$. On the other hand, the behaviors of these functions are different each other around $p_i = 1$ and $p_i = 0$.

4 Software Reliability Assessment Measures

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

$$\begin{aligned}\mathrm{E}[N(i)] &= \sum_{z=0}^{n} z \sum_{n} \binom{n}{z} \{P(i)\}^z \{1-P(i)\}^{n-z} Pr\{N_0 = n\} \\ &= \mathrm{E}[N_0]P(i).\end{aligned} \tag{8}$$

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

$$\begin{aligned}\mathrm{Var}[N(i)] &= \mathrm{E}[N(i)^2] - (\mathrm{E}[N(i)])^2 \\ &= \mathrm{Var}[N_0]\{P(i)\}^2 + \mathrm{E}[N_0]P(i)\{1-P(i)\}.\end{aligned} \tag{9}$$

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 continued 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

$$\begin{aligned}R(i,h) &= \sum_{k} \Pr\{N(i+h) = k \mid N(i) = k\}\Pr\{N(i) = k\} \\ &= \sum_{k} \Big[\{P(i)\}^k \{1 - P(i+h)\}^{-k} \cdot \sum_{n} \binom{n}{k} \{1 - P(i+h)\}^n \cdot \Pr\{N_0 = n\}\Big].\end{aligned} \tag{10}$$

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

$$R(i,h) = [1 - \lambda\{P(i+h) - P(i)\}]^K. \tag{11}$$

Furthermore, discrete instantaneous and cumulative mean time between software failures (MTBFs), $MTBF_I(i)$ and $MTBF_C(i)$, are also derived as

$$MTBF_I(i) = 1\Big/(\mathrm{E}[N(i+1)] - \mathrm{E}[N(i)]), \tag{12}$$

$$MTBF_C(i) = i\Big/\mathrm{E}[N(i)], \tag{13}$$

respectively.

5 Parameter Estimation Method

We compare the performance of our models for software reliability assessment with the existing corresponding model, which does not consider the effect of test environment 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 measured on the basis of weeks and program size $K = 1.630 \times 10^5$ (LOC) [3],

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 measured on the basis of weeks and 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 comparison we treat the C0 testing coverage as the test environment factors affecting the software failure occurrence or fault detection phenomenon. Thus, we assume $\boldsymbol{\beta}_i \equiv c_i$. Regarding the actual data, DS1 shows an exponential software reliability growth curve and DS2 shows an S-shaped curve. 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), 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 errors (abbreviated as MSE) [10], which is calculated as

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}[y_k - \widehat{\mathrm{E}}[N(t_k)]]^2. \tag{14}$$

The MSE represents the mean square errors of the number of detected faults between the estimated and actual values for all of the observed data points. Table 1 shows the results of model comparisons based on the MSE. From Table 1, we can say our models keep high fitting performance to the actual data even though the actual data shows an exponential or S-shaped software reliability growth curve. Furthermore, we conducted a goodness-of-comparison based on the Akaike information criterion (AIC) [1] because a smaller MSE is not sufficient to conclude that our approach is better than the existing corresponding model and the degree of freedom of our models became higher. The AIC is calculated as

$$\mathrm{AIC} = -2\mathrm{MLL} + 2\phi, \tag{15}$$

where MLL represents the maximum log likelihood and ϕ indicates the number of parameter. We judged that a model indicating smaller AIC fits better to the actual data. Table 1 also shows the results of model comparisons based on the MLL and AIC. From the results of model comparisons based on the AIC in Table 1, it can be said that our models possess better fitting performance even in the case considering the degree of freedom of our models.

We show examples of the application of software reliability analysis based on our model by using the actual data set DS2. Especially, we show the numerical examples in case that p_i follows the probit

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

		MSE	MLL	AIC
DS1	logistic	27.308	-2240.31	4488.62
	probit	26.958	-2240.27	4499.534
	complementaly log-log	31.105	-2240.70	4489.40
	Existing	28.256	-2251.59	4509.17
DS2	logistic	33.954	-3052.13	6112.26
	probit	33.475	-3052.07	6112.14
	complementary log-log	34.151	-3052.13	6112.27
	Existing model	39.713	-3054.97	6115.94

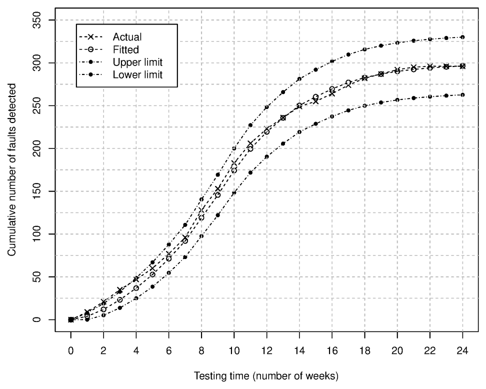

Fig 1 : Estimated expected number of faults detected, $\widehat{\mathrm{E}}[N(i)]$, and the 95% confidence limits for DS2.

regression approach. We obtain the parameter estimates by the method of maximum likelihood as $\widehat{\lambda} = 1.5167 \times 10^{-3}$, $\widehat{\alpha}_0 = -2.2289$, $\widehat{\alpha}_1 = 3.9569 \times 10^{-1}$, and $\widehat{\gamma} = 1.5755$, where $\widehat{\lambda}$, $\widehat{\alpha}_0$, $\widehat{\alpha}_1$, and $\widehat{\gamma}$ are the parameter estimates of λ, α_0, α_1, and γ, respectively. Then, the expected number of initial faults can be estimated as $K \times \widehat{\lambda} \simeq 298$ because $K = 1.972 \times 10^5$ (LOC) in DS2. Figure 1 depicts the estimated time-dependent behavior of the expected number of faults detected, $\widehat{\mathrm{E}}[N(i)]$, and its 95% confidence limits. The 100γ% confidence limits for $\widehat{\mathrm{E}}[N(i)]$ are derived as $\widehat{\mathrm{E}}[N(i)] \pm K_\gamma \sqrt{\widehat{\mathrm{Var}}[N(i)]}$, where K_γ indicates the $100(1+\gamma)/2$ percent point of the standard normal distribution [12].

6 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 [7, 9, 11] by using a lot of software fault-counting data with software metrics.

Acknowledgement

This research was supported in part by the Grant-in-Aid for Scientific Research (C), Grant No. 16K00098, from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] H. Akaike, "A new look at the statistical model identification,"*IEEE Trans. Autom. 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,"*Math. Comput. 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] S. Inoue and S. Yamada, "Generalized discrete software reliability modeling with effect of program size," *IEEE Trans. Syst. Man Cybern. — Part A: Syst. Hum.*, Vol. 37, No. 2, pp. 170–179, 2007.

[5] S. Inoue and S. Yamada, "Discrete program-size dependent software reliability assessment: modeling, estimation, and goodness-of-fit comparisons," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, Vol. E90-A, No. 12, pp. 2891–2902, 2007.

[6] M. Kimura, S. Yamada, H. Tanaka and S. Osaki, "Software reliability measurement with prior-information on initial fault content," *Trans. Inf. Process. Soc. Jpn.*, Vol. 34, No. 7, pp. 1601–1609, 1993.

[7] 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.

[8] T. Nakagawa and S. Osaki, "The discrete Weibull distribution," *IEEE Trans. Reliab.*, Vol. R-24, No. 5, pp. 300–301, 1975.

[9] 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.

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

[11] 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.

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