Journal of Universal Computer Science, vol. 24, no. 12 (2018), 1680-1689
submitted: 26/1/18, accepted: 15/11/18, appeared: 28/12/18 © J.UCS

A Software Reliability Growth Modeling Framework with Complexity of Path Searching

Shinji Inoue
(Kansai University, Takatsuki, Osaka
ino@kansai-u.ac.jp)

Shigeru Yamada
(Tottori University, Tottori, Tottori
yamada@tottori-u.ac.jp)

Abstract: We discuss measurement method for software system with the complexity of path searching based on an infinite server queueing theory. Our modeling framework describes concretely the process of the software fault detection by representing on the infinite server queueing system. Additionally, we discuss estimation method of parameters is our model, and derive a few measures for reliability assessment of software systems. Further, we derive several software reliability growth models from our framework. Finally, numerical illustrations based on our specific model, which is obtained from our modeling framework, are given by using actual fault counting-data.

Keywords: Reliability assessment, software systems, test cases, path searching time, infinite server system, generalized modeling framework
Categories: D.2

1 Introduction

Reviewing activities of each development phase and testing activities in the software development process is a key factor for ensuring the quality/reliability of software systems. Needless to say, software quality management technologies also important for managing the quality of the process. Among the software quality characteristics, we cannot ignore the reliability because the reliability is regarded as so-called must-be quality. Especially, quantitative measurement, assessment and prediction of the software quality/reliability are important for checking quantitatively the quality. A software reliability growth models (abbreviated as SRGMs) [Pham 2000, Yamada 2014] is known as a mathematical model. The SRGMs are useful for such quality/reliability assessment activities.

Recently, generalized modeling frameworks have been discussing by many researchers for unifying SRGMs proposed so far. Dohi et al. [Dohi et al. 1998] proposed a generalized modeling frameworks for SRGMs following nonhomogeneous Poisson processes (NHPP), so-called NHPP models, by describing software detection process in an infinite server queueing theory [Ross 1992, Trivedi 2002]. Huang et al. [Huang et al. 2003] have discussed a unified scheme for the discrete-time models by focusing on the several mathematical structures. Inoue and Yamada [Inoue and Yamada 2006] proposed an extended delayed S-shaped models

by describing the consecutive software failure-occurrence and fault-removing processes based on the infinite server queueing theory. Further, Inoue and Yamada [Inoue and Yamada 2007] have proposed generalized discrete software reliability modeling framework with program size by using the generalized order statistics theory [Langberg and Singpurwalla 1985].

This paper focuses on the literature, Dohi et al. [Dohi et al. 1998]. They developed an infinite server queueing system describing consecutive the test case execution (arrival) process and the path searching time of each test execution. And they assumed that a path searching time of a test-case execution follows an independent and identically distributed random time. Further the number of observed faults are regarded as the number of program paths not completed searching. Considering an actual situation, the path searching time must be influenced from the complexity of the program paths to be executed by test cases. This paper develops infinite server queueing system for software reliability measurement with the complexity of the path searching. Further, we propose a few specific SRGMs following our modeling frameworks. And we demonstrate reliability analysis based on our specific proposed models by actual data collected in a software testing phase.

2 Infinite Server Queueing Model

A modeling framework proposed so far [Dohi et al. 1998] describes the process from the arrival of test-cases to the software failure-occurrence by using the infinite server queueing system assumed:

(A1) Test cases have been completely prepared before the beginning of testing activities, and are executed by following homogeneous Poisson process (HPP) with mean λ

(A2) The i-th test case has the data for executing X_i program path.

(A3) The each path searching time T follows the distribution function $S(t) \equiv \Pr\{T \leq t\}$ (i.i.d.).

We introduce the counting process $\{Z(t), t \geq 0\}$ representing the number of faults detected up to testing time t. Treating the number of detected faults as the number of program paths still executing at testing time t, we have

$$\Pr\{Z(t) = z\} = \sum_{j=0}^{\infty} \exp[-\lambda t] \frac{(\lambda t)^j}{j} \times \Pr\{Z(t) = z | X_1 + X_2 + \cdots + X_j = x\}. \tag{1}$$

Eq. (1) implies the stochastic process follows a compound Poisson process with parameter λ and jump size X_i. Now we assume that the one test-case has the data for executing one program path and one program path has one software fault at most, i.e., $X_i = 1$ $(i = 1, 2, \cdots)$. At this time, the process $\{Z(t), t \geq 0\}$ is recognized as the $M/G/\infty$ queueing system. Then, we can obtain

$$\Pr\{Z(t) = z\} = \frac{[\lambda \int_0^t \bar{S}(t)dt]^z}{z!} \exp\left[-\lambda \int_0^t \bar{S}(t)dt\right], \tag{2}$$

which can be treated as the NHPP with mean $\lambda \int_0^t \bar{S}(t)dt$. In Eq. (2),$\bar{S}(t) \equiv 1 - S(t)$. From Eq. (2), we can obtain several types of NHPP models by assuming specific suitable path searching time distributions in Eq. (2).

3 Extended Infinite Server Queueing Model with Complexity of Path Searching

We can easily consider that the stochastic property of the path searching time depends on the complexity of program path. Therefore, we extend the infinite server queueing system discussed in Section 2 by considering with the difference of the path searching time. Our infinite server queueing system is developed by the following assumptions:

(B1) Test cases have been completely prepared before the beginning of testing activities, and are executed by following HPP with mean λ.

(B2) One test case has the data for executing one program path and one program path has one fault at most.

(B3) A test case for the low complexity program path is executed with probability p. A test case for the high complexity program path is executed with probability $1 - p$.

(B4) The random variables for the path searching time are independently distributed. However, the path searching time for the low and high complexity paths follow probability distribution functions $J(t)$ and $K(t)$, respectively.

Following the notion of generalized modeling framework in Eq. (1), we obtain Figure 1 depicting our extended infinite server queueing system.

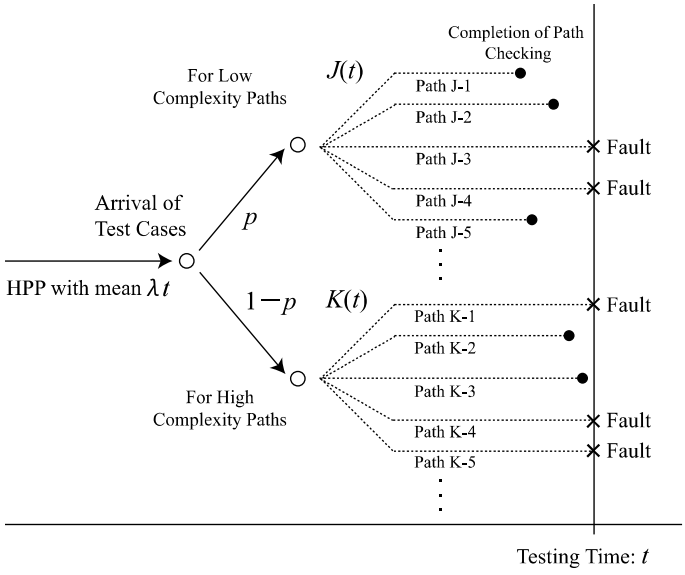

Figure 1: Illustration of our infinite server queueing system.

Let $\{Y(t), t \geq 0\}$ and $\{Z(t), t \geq 0\}$ denote the numbers of test case executed and faults detected up to testing time t, respectively. Further we introduce $\{Y_1(t), t \geq 0\}$ and $\{Y_2(t), t \geq 0\}$ are the stochastic processes representing the number of test cases for the low and high complexity program paths executed up to time t, respectively. And $\{Z_1(t), t \geq 0\}$ and $\{Z_2(t), t \geq 0\}$ represents the number of faults counted from the low and high complexity program paths executed during $(0, t]$, respectively. Using the above stochastic processes, we formulate the infinite server queueing system shown in Fig. 1.

Now we analyze the software fault-detection phenomenon occurred by the execution of the low complexity program paths. First of all, $\{Y_1(t), t \geq 0\}$ is formulated as

$$\begin{aligned} \Pr\{Y_1(t) = c\} &= \sum_{d=0}^{\infty} \Pr\{Y_1(t) = c, Y_2(t) = d\} \\ &= \frac{(p\lambda t)^c}{c!} \exp[-p\lambda t], \end{aligned} \tag{3}$$

from the modeling assumptions. Eq. (3) implies that the test cases for the low complexity program paths are executed in accordance with a HPP with parameterλp. Then, we can obtain $\{Z_1(t), t \geq 0\}$$ as

$$\Pr\{Z_1(t) = m\} = \sum_{j=0}^{\infty} \Pr\{Z_1(t) = m |\, Y_1(t) = j\} \frac{(p\lambda t)^j}{j!} \exp[-p\lambda t], \tag{4}$$

The conditional probability, $Pr\{N_1(t) = m | X_1(t) = j\}$, in Eq. (4) means that m faults are detected from j executed low complexity program paths. Therefore, we give the conditional probability as

$$\Pr\{Z_1(t) = m | Y_1(t) = j\} = \begin{cases} \binom{j}{m} \{d(t)\}^m \{1 - d(t)\}^{j-m} & (m \leq j), \\ 0 & (m > j), \end{cases} \tag{5}$$

where $d(t)$ is the probability that the path searching is not completed during $(0, t]$ given that the corresponding test case has been executed up to testing time x $(< t)$. Then, $r(t)$ is given as

$$r(t) = \int_0^t \frac{\bar{J}(t - x)}{t} dx \tag{6}$$

by the concepts of the conditional time distribution [Ross 1992]. Consequently, we obtain

$$\Pr\{Z_1(t) = m\} = \frac{\{M_1(t)\}^m}{m!} \exp[-M_1(t)], \tag{7}$$

where

$$M_1(t) = p\lambda \int_0^t \bar{J}(\tau) d\tau. \tag{8}$$

As far as the software fault-detection phenomenon for the high complexity program paths, we can analyze $\{Z_2(t), t \geq 0\}$ as:

$$M_2(t) = (1-p)\lambda \int_0^t \bar{K}(\tau)d\tau. \tag{9}$$

by applying the same methodology in deriving $\{Z_1(t), t \geq 0\}$ mentioned above. From the above discussions, we can obtain $\{Z(t), t \geq 0\}$ as

$$\begin{aligned} \Pr\{Z(t) = n\} &= \Pr\{Z_1(t) + Z_2(t) = n\} \\ &= \frac{\{M_1(t) + M_2(t)\}^n}{n!} \exp[-\{M_1(t) + M_2(t)\}], \end{aligned} \tag{10}$$

which is an NHPP with mean $\{M_1(t) + M_2(t)$. We can see that we can obtain several NHPP models with the complexity of path searching by using Eq. (10) by giving suitable probability distribution functions of $J(t)$ and $K(t)$, respectively.

4 Specific NHPP Models

We show specific NHPP models by following the framework discussed in Section 3. For examples, we develop three types of new NHPP models by assuming the path searching time distributions, $J(t)$ and $K(t)$, which are for the low and high complexity program paths, respectively

4.1 Model 1

As Model 1, we assume that

$$J(t) = 1 - \exp[-\alpha t] \quad (\alpha > 0), \tag{11}$$

$$K(t) = 1 - \exp[-\beta t] \quad (\alpha > \beta > 0), \tag{12}$$

respectively. Eqs. (11) and (12) mean that the expected searching time for the high complexity program paths is longer than the low complexity program paths. From Eqs. (11) and (12), the mean of Eq. (10) is derived as

$$H(t) = \lambda \left[\frac{p}{\alpha}(1 - e^{-\alpha t}) + \frac{1-p}{\beta}\left(1 - e^{-\beta t}\right)\right]. \tag{13}$$

4.2 Model 2

As Model 2, we assume $J(t)$ and $K(t)$ follows the following exponential distribution with parameters α (> 0) and Rayleigh distribution with parameter β (> 0):

$$J(t) = 1 - \exp[-\alpha t] \quad (\alpha > 0), \tag{14}$$

$$K(t) = 1 - \exp\left[-\left(\frac{t}{\beta}\right)^2\right] \quad (\beta > 0), \tag{15}$$

respectively. In this case, the difference of the stochastic properties for path searching time is represented by the shapes of the distributions. From Eqs. (14) and (15), the mean value function of Eq. (10) is derived as

$$H(t) = \lambda\left[\frac{p}{\alpha}(1-e^{-\alpha t}) + \frac{\beta(1-p)}{2}\left\{\Gamma_1\left(\frac{1}{2}\right) - \Gamma_2\left(\frac{1}{2}, \left(\frac{t}{\beta}\right)^2\right)\right\}\right], \tag{16}$$

where $\Gamma_1(k)$ and $\Gamma_2(k,m)$ are the complete and incomplete gamma distributions,

$$\Gamma_1(k) = \int_0^\infty e^{-x}x^{k-1}dx, \tag{17}$$

$$\Gamma_2(k,m) = \int_m^\infty e^{-x}x^{k-1}dx, \tag{18}$$

respectively. We call Eq. (16) Model 2.

4.3 Model 3

In Model 3, assuming $J(t)$ and $K(t)$ obey the following exponential distribution with parameter α (>0) and gamma distribution with $k=2$ (shape parameter), we have

$$J(t) = 1-\exp[-\alpha t] \quad (\alpha > 0), \tag{19}$$

$$K(t) = 1-(1+\beta t)\exp[-\beta t] \quad (\beta > 0), \tag{20}$$

Then, the mean of Eq. (10) can be obtained as

$$H(t) = \lambda\left[\frac{p}{\alpha}(1-e^{-\alpha t}) + \frac{2(1-p)}{2}\left\{1-\left(1+\frac{\beta}{2}t\right)e^{-\beta t}\right\}\right]. \tag{21}$$

5 Model Parameter Estimation

Following the method of maximum-likelihood based on observed fault-counting data, we obtain the values of parameters. However, it might not be appropriate to obtain the value of the parameter p, the probability of the execution of the low complexity program paths, from the fault-counting data. Therefore, we consider that $p=\bar{p}$ has been given in this parameter estimation. Supposing K data pairs: $(t_k, y_k)(k=0,1,2,\cdots,K)$ has been observed. y_k is the total number of faults detected during $(0,t_k]$ and $\bar{p}$. Then, we obtain $\ln L(\boldsymbol{\theta}|\bar{p})$, which is the log-likelihood function for the counting process $\{Z(t), t\geq 0\}$. $\ln L(\boldsymbol{\theta}|\bar{p})$ represents the log-likelihood function given $\bar{p}$, $\boldsymbol{\theta}$ is the set of parameters in the model. The values of parameters can be estimated by numerically solving the equations:

$$\frac{\partial \ln L(\boldsymbol{\theta}|\bar{p})}{\partial \boldsymbol{\theta}} = 0, \tag{22}$$

simultaneously in terms of each parameter of the model.

6 Software Reliability Analysis

We use fault counting data, $(t_k, y_k)(k = 0, 1, 2, \cdots, 36\ (\text{days}))$, which is observed in actual software testing for embedded control software of printer system, which consists of main and five sub components.

First of all, we need to obtain the value of parameter $\bar{p}$ from the information of the prepared test cases. In our numerical examples, we regard the test cases executing within only the main component as the test cases for the low complexity program paths. And we also regard the test cases executing through both of the main and sub components as the test cases for the high complexity program paths. From the information of the prepared test cases, we calculate $\bar{p} = 0.288$. We show software reliability analysis by using Model 2 in Eq. (16) and Model 3 in Eq. (21), respectively.

We show the examples in case of applying Model 2 in Eq. (16) to the observed data. Model 2 has the following three parameters, λ, α, and β, which are needed to estimate from the observed data. Consequently, we estimate

$\hat{\lambda} = 2.034$, $\hat{\alpha} = 0.158$ and $\hat{\beta} = 0.870 \times 10^2$, where $\hat{\lambda}$, $\hat{\alpha}$, and $\hat{\beta}$ represents the parameter estimations of λ, α, and β, respectively. In Figure 2, we show the estimated $\widehat{H}(t)$ (Model 2), and its 90% confidence limits by using the estimated parameter estimations $\hat{\lambda}$, $\hat{\alpha}$, and $\hat{\beta}$. The $100p$ confidence limits of $E[N(t)] \equiv H(t)$ is obtained by

$$\widehat{H}(t) \pm U_p \sqrt{\widehat{H}(t)}, \tag{23}$$

where U_p is the $100(1+p)/2$ quantile of the standard normal distribution [Yamada and Osaki 1985]. The confidence limits might be useful for the process control of the testing phase.

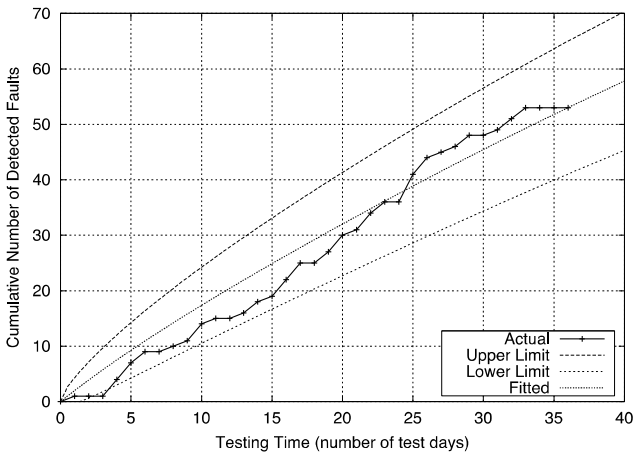

Figure 2: Estimated time-dependent behavior of $\widehat{H}(t)$, and its 90% confidence limits. (Model 2; $\bar{p} = 0.288$)

Further, $R(x|t)$ represents the failure-free operation probability during the time interval $(t, t+x](x \geq 0, t \geq 0)$ assuming the software system is operated as the same environment as the testing environment time interval $(0, t]$. Then, the functions is obtained as

$$R(x|t) = \exp[-\{H(t+x) - H(t)\}], \tag{24}$$

by using the stochastic properties of NHPP. Figure 3 shows the estimated software reliability function $\hat{R}(x|36)$. In Fig. 3, we can obtain $\hat{R}(1.0|36) \approx 0.296$.

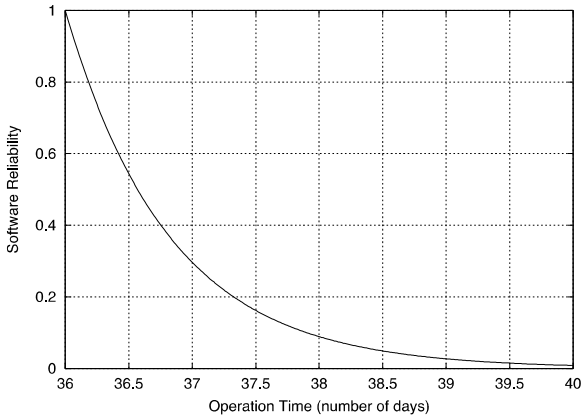

Figure 3: Estimated $\hat{R}(x|36)$. *(Model 2;* $\bar{p} = 0.288$*)*

Furthermore, we show numerical examples for Model 3 in Eq. (21). There exists the three parameters, λ, α and β, in Model 3. Conducting parameter estimation, we obtain $\hat{\lambda} = 2.000$, $\hat{\alpha} = 1.663 \times 10^{-1}$ and $\hat{\beta} = 1.414 \times 10^{-2}$, respectively. Figure 4 shows the estimated $\widehat{H}(t)$ in Model 3 and its 95% confidence limits. And the estimated $\hat{R}(x|36)$, which is derived from Model 3, is shown in Figure 5. we obtain $\hat{R}(1.0|36)$ as 0.275 from Fig. 5.

7 Conclusions

We discussed reliability assessment of software systems by considering the difference of the stochastic property of the program path searching time based on the infinite server queueing theory. Several models are developed by considering the stochastic behavior of the path searching time of the low and high complexity program paths. Further we show application examples of our modeling framework by developing specific models, and reliability analysis based on them by observed actual data. Further studies are still needed to discuss more on the goodness-of-fit of models developed by following proposed framework, and challenge to reflect more on actual software fault-detection process.

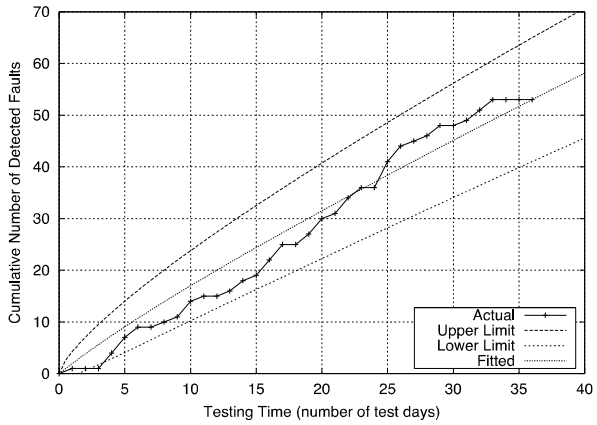

Figure 4: Estimated time-dependent behavior of $\hat{H}(t)$, and its 90% confidence limits. (Model 3; $\bar{p} = 0.288$)

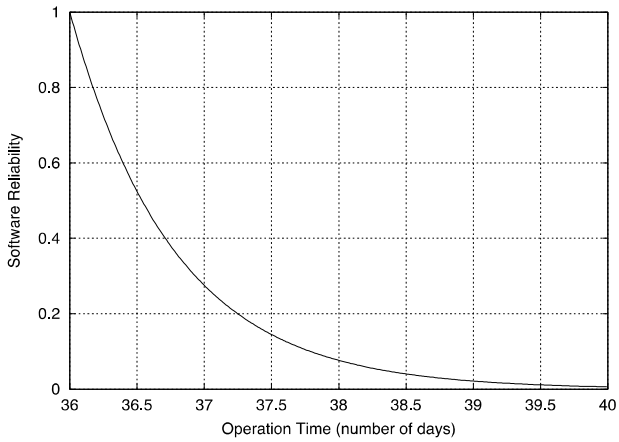

Figure 5: Estimated $\hat{R}(x|36)$. (Model 3; $\bar{p} = 0.288$)

References

[Dohi et al. 1998] Dohi, T., Matsuoka, T., Osaki, S.: ``An infinite server queueing model for assessment of the software reliability"; Elect. Comm. in Japan (Part 3), 85, 3 (2000) 536-544.

[Huang et al. 2003] Huang, C.Y., Lyu, M.R., Kuo, S.Y.: ``A unified scheme of some nonhomogeneous Poisson process models for software reliability estimation"; IEEE Trans. Sotw. Eng., 29, 1 (2003) 261-269.

[Inoue and Yamada 2006] Inoue, S., Yamada, S.: "An extended delayed S-shaped software reliability growth model based on infinite server queueing theory"; in Reliability Modeling, Analysis and Optimization, Pham, H. Ed. 357-372, World Scientific, Singapore 2006.

[Inoue and Yamada 2007] Inoue, S., Yamada, S.: `"Generalized discrete software reliability modeling with effect of program size"; IEEE Trans. Syst. Man, Cybern. (Part A), 37, 2 (2007) 170-179.

[Langberg and Singpurwalla 1985] Langberg, N., Singpurwalla, N.D.: "A unification of some software reliability models"; SIAM J. Sci. Stat. Comput., 6, 3 (1985) 781-790.

[Pham 2000] Pham, H.: "Software Reliability"; Springer-Verlag, Singapore, 2001.

[Ross 1992] Ross, S.M.: "Applied Probability Models with Optimization Applications"; Dover Publications, New York, 1992.

[Trivedi 2002] Trivedi, K.S.: "Probability and Statistics with Reliability, Queueing and Computer Science"; John Wiley & Sons, New York, 2002.

[Yamada and Osaki 1985] Yamada, S., Osaki, S.: "Software reliability growth modeling: Models and applications"; IEEE Trans. Softw. Eng., SE-11, 12 (1985) 1431-1437.

[Yamada 2014] Yamada, S.: "Software Reliability Modeling: Fundamentals and Applications"; Springer-Verlag (Springer Briefs in Statistics), Tokyo/Heidelberg, 2014.