International Journal of Mathematical, Engineering and Management Sciences
Vol. 5, No. 1, 76-82, 2020
https://doi.org/10.33889/IJMEMS.2020.5.1.007

Phase-Type Modeling Approaches for Software Reliability Modeling with Debugging Process

Shinji Inoue
Faculty of Informatics,
Kansai University, Osaka, Japan.
Corresponding author: ino@kansai-u.ac.jp

Shigeru Yamada
Graduate School of Engineering,
Tottori University, Tottori, Japan.
E-mail: yamada@tottori-u.ac.jp

(Received March 21, 2019; Accepted August 29, 2019)

Abstract
Reflecting the software fault debugging procedure or environment of testing activities on software reliability models is often discussed as the approaches for improving assessment accuracy for model-based reliability assessment. We discuss a modeling approach reflecting software debugging procedure based on phase-type modeling scheme and propose probability models for software reliability measurement. Further, we give brief consideration for the usefulness of this modeling approach by using a few data sets.

Keywords- Software debugging process, Phase-type probability distribution, Software reliability assessment, Mathematical model.

1. Introduction
For conducting estimation of the failure-free operation probability of software system and other useful reliability assessment measures, it is known software reliability models (Yamada, 2014) is utilized practically in reliability assessment of software system. Actually, a lot of discussions on software reliability modeling by considering actual testing environment for improving the quality of model-based assessment of software reliability. For examples, discretization of continuous time models and discretized models, which has been derived by discretizing the continuous-time models, have been focused as one of the approaches for considering software fault-counting data collection activities (Inoue and Yamada, 2006). And change-point software reliability modeling approaches have been proposed by considering the change of testing environment and the influence on stochastic behavior of reliability growth process during the testing phase (Inoue and Yamada, 2015). Considering other factors, such as software fault debugging process and software complexity, must be useful for developing more useful software reliability models which enables us to reliability assessment with increased accuracy (Inoue and Yamada, 2007). Recently, as one of the approaches for improving the quality of assessment and for yielding unified modeling scheme, a phase-type modeling approach (Okamura and Dohi, 2016) has been proposed. In this modeling approach, the perfect fault-correction time follows a phase-type probability distribution describing the uncertainty of the time to absorption in a CTMC, which is the abbreviation of a continuous-time Markov chain. It is known several probability distribution functions are described by considering the counterpart to the absorbing CTMC. However, we need to assume the appropriate continuous-time Markov chain representing possible situation of software debugging

process when we obtain a software reliability model by following this modeling approach.

This paper gives one of the solutions on applying the phase-type approach for reflecting actual testing environment in software reliability assessment. That is, we apply this probability distribution to describing the uncertainty of the fault elimination procedure in the test activities. Then, we give several discussions on how to apply the phase-type modeling scheme to a software debugging procedures by considering several types of possible software debugging processes. This paper is expected to contribute to applying practically the phase-type modeling scheme for developing useful software reliability models reflecting actual debugging process.

2. Modeling Framework

In our discussion, we assume the following situations for developing a mathematical model in software reliability assessment (Langberg and Singpurwalla, 1985):

(A1) Z_0 (> 0) faults have been introduced before testing. And Z_0 (> 0) is a random variable.
(A2) Successive software failure observation and its perfect fault-correction follows an i.i.d. probability distribution function $G_{PH}(t)$, which is a phase-type probability distribution function.
(A3) We do not consider any fault introduction during the debugging process.

From the assumptions above, we formulate the time dependent uncertainty for the number of detected faults, which is denoted by $\{Z(t), t \geq 0\}$, as follows:

$$\begin{aligned}\Pr\{Z(t) = k\} &= \sum_{n} \binom{n}{k} \{G_{PH}(t)\}^k \{1 - G_{PH}(t)\}^{n-k} \times \Pr\{Z_0 = n\} \\ &= \frac{\{\omega G_{PH}(t)\}^k}{k!} \exp[-\omega G_{PH}(t)],\end{aligned} \tag{1}$$

under the condition that Z_0 is a Poisson random variable taking mean ω (> 0). From Eq. (1), it is possible to obtain the failure free probability in operation during the time-interval $(t, t + x]$$(x \geq 0)$, for example.

2. Phase-Type Modeling Approach

The phase-type distribution (Buchholz et al., 2014) describes the uncertainty of the time by considering an absorbing CTMC. Considering an absorbing CTMC with the set of transient states $\mathcal{S}_T = \{1, 2, \cdots, n\}$ and the absorption $\mathcal{S}_A = \{n + 1\}$, we can characterize the behavior of the absorbing CTMC by the infinitesimal generator $\boldsymbol{A}$ shown as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B_0} & \boldsymbol{b_1} \\ \boldsymbol{0} & 0 \end{pmatrix}. \tag{2}$$

In Eq. (2), $n \times n$ submatrix $\boldsymbol{B_0}$ represents the transition rates within the transient states. The transition rates to the absorption from the transient states are expressed by $\boldsymbol{b_1}$, $n \times 1$ vector. There is no transition from the absorption by the row vector $\boldsymbol{0}$ in Eq. (2). Analyzing the CTMC, we can formulate the uncertainty of the time to reach the absorption from transient states as

$$\begin{aligned} G_{PH}(t) &= \Pr\{T \le t\} \\ &= 1 - \boldsymbol{\pi} \exp[\boldsymbol{B_0 t}]\,\mathbf{1} \qquad (x \ge 0). \end{aligned} \tag{3}$$

In Eq. (3), $\boldsymbol{\pi}$ is the initial state vector, $\mathbf{1}$ is the column vector in which the all elements are 1.

We discuss descriptions of a software debugging process, and also discuss the relationship to the existing models. Now, we consider the absorbing CTMC shown in Figure 1. Actually, Figure 1 depicts the state transition diagram for the debugging situation of existing well-known exponential software reliability growth model (Musa et al., 1987). In Figure 1, the debugging process consists of just one process, which is the software failure detection process, denoted by (P1). Further this model involves the assumption that the fault is immediately and perfectly debugged. From Figure 1, we obtain

$$\boldsymbol{A} = \begin{pmatrix} -z & z \\ 0 & 0 \end{pmatrix}, \tag{4}$$

and $\boldsymbol{\pi} = (1)$, respectively. Then, $G_{PH}(t)$ in Eq. (1) follows the exponential distribution with mean z from Eq. (3). Consequently, an exponential model: $M_e(t) = \omega(1 - \exp[-zt])$ can be obtained by following the modeling framework shown in Eq. (1). Further we give another explanation of our notion in describing debugging process by analyzing a delayed S-shaped model (Yamada, 2014). This model assumes the consecutive failure detection and the fault removal debugging processes. Based on the assumption, we obtain

$$\boldsymbol{A} = \begin{pmatrix} -z & z & 0 \\ 0 & -z & z \\ 0 & 0 & 0 \end{pmatrix}, \tag{5}$$

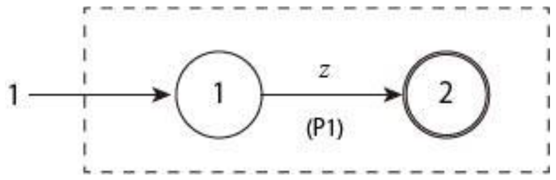

Figure 1. Phase structure for equation (4)

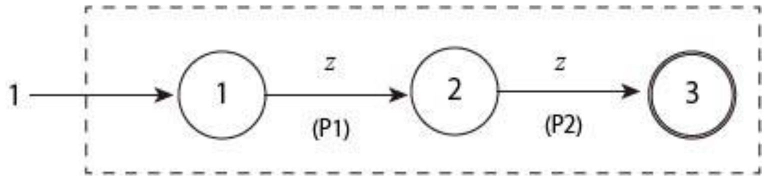

Figure 2. Phase structure for equation (5)

and $\boldsymbol{\pi} = (0 \quad 1)$, respectively. Figure 2 shows the absorbing CMTC representing the debugging process in the delayed S-shaped software reliability growth model. Following Eq. (3), the phase-type distribution can be derived. Consequently, we obtain $M_d(t) = \omega\{1 - (1 + zt)\exp[-zt]\}$ as the mean value function in Eq. (1).

3. Our Models Considering Debugging Process Scenario

We consider several kinds of possible debugging process in practical debugging situation and its difficulties on the software detection and removal. And we show how to apply the phase-type modeling scheme for obtaining a useful software reliability model for developing a stochastic model in software reliability assessment.

Now we consider the case of Figure 3 representing a debugging process. The debugging process in Figure 3 consists of the following three debugging processes: failure-detection (P1), cause analysis

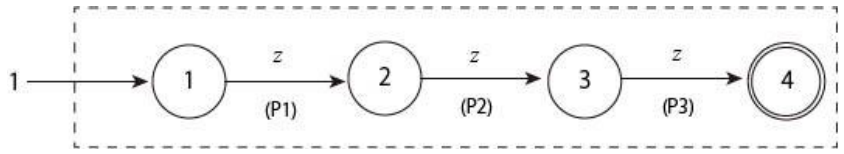

Figure 3. Phase structure for the software debugging process in Model A

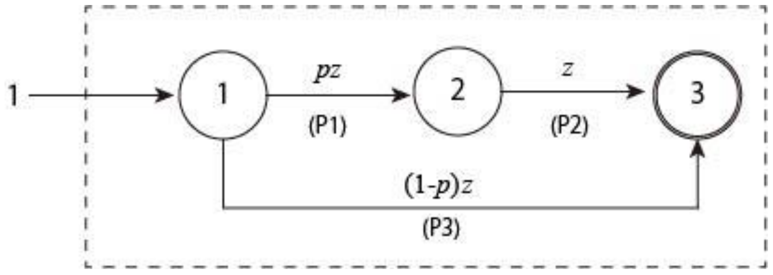

Figure 4. Phase structure for the software debugging process in Model B

(P2), fault removal (P3). From Figure 3, we have

$$\boldsymbol{A} = \begin{pmatrix} -z & z & & \\ & -z & z & \\ & & -z & z \\ & & & \end{pmatrix}, \tag{6}$$

where the null elements represent 0. We also obtain $\boldsymbol{\pi} = (1 \quad 0 \quad 0)$. Then, we obtain $G_{PH}(t)$ for Figure 3 as

$$G_{PH}(t) = 1 - \boldsymbol{\pi} \exp\left[\begin{pmatrix} -z & z & 0 \\ 0 & -z & z \\ 0 & 0 & -z \end{pmatrix} x\right]\mathbf{1}$$
$$= 1 - \left\{1 + zx + \frac{1}{2}(zx)^2\right\}\exp[-zx]. \tag{7}$$

Consequently, we have

$$M_1(t) = \omega\left[1 - \left\{1 + zx + \frac{1}{2}(zx)^2\right\}\exp[-zx]\right], \tag{8}$$

by Eq. (1). We call Eq. (8) “Model A”.

Considering the difficulties on software fault debugging, we propose another model. This model is a hybrid of the debugging processes in Figures 2 and 3. Figure 4 shows the phase structure for the debugging process of this model. In this model, we consider high and low classification in software debugging procedure. Concretely, software failures classified into the high difficulty and the faults are detected and removed through the failure detection (P1) and fault removal (P2) with probability p. On the other hand, the failures classified into the low difficulty are removed immediately by the fault removal (P3) with probability $(1-p)$. From Figure 4, we obtain

Table 1. Results of goodness-of-fit comparisons

Data	Model	MSE	MLL	AIC
D1	Delayed S-shaped	36.650	-68.191	140.38
	Model A	99.354	-82.431	168.86
	Model B	**26.332**	**-66.326**	**136.65**
D2	Delayed S-shaped	167.77	-94.604	193.21
	Model A	288.12	-144.11	292.21
	Model B	**35.547**	**-57.219**	**118.44**
D3	Delayed S-shaped	188.75	-109.19	222.38
	Model A	394.08	-138.73	281.46
	Model B	**130.58**	**-102.20**	**208.39**

$$\boldsymbol{A} = \begin{pmatrix} -z & pz & (1-p)z \\ & -z & z \\ & & \end{pmatrix}, \tag{9}$$

where the null elements represent 0. And the initial state vector is obtained as $\boldsymbol{\pi} = (0 \quad 1)$. Then, the mean value function is

$$M_2(t) = \omega\{1 - (1 + pzt)\exp[-zx]\}. \tag{10}$$

We call Eq. (10) “Model B”.

4. Comparisons with Existing Model

We do comparisons of our models and existing delayed S-shaped model (Delayed S-shaped) (Yamada, 2014), which has been introduced in Section 2, in terms of the MSE, MLL, and AIC (Yamada, 2014). The MLL means the maximum likelihood. Now we apply the following three fault count data: D1, D2, and D3 (Inoue and Yamada, 2006). The fault-counting data D1 and DS3 show S-shaped curves and D2 shows an exponential growth curve, respectively.

In Table 1 the best values in terms of each criterion show by using the bold fonts. We can say our Model B is expected to show better fitting performance to the observed data than other models from Table 1. From these results, we can see the importance of focusing on the debugging procedure when we do software reliability modeling. And we can see that the usefulness of the phase-type modeling scheme in describing several kinds of debugging situations which must be observed in actual testing activities.

5. Conclusion

We discussed how to apply the phase-type modeling scheme for developing a specific software reliability model. Concretely, we applied this modeling scheme to describing the uncertainty of the software debugging procedures. Further, we developed new types of software reliability models reflecting a possible software debugging procedures with the difficulty in debugging activities. As we discussed, we can say that the phase-type modeling scheme is expected to yield some other models in software reliability assessment by describing a lot of kinds of software debugging procedures and physical meanings in debugging activities. In model comparisons, we recognize the software debugging process-oriented approach described by the phase-type probability distribution is expected to obtain more plausible models for software reliability assessment. However, we need more investigations for making sure the usefulness of modeling approach discussed in this paper. And we are interested in seeking some other models developed by following this modeling approach.

Conflict of Interest
The authors declare that there is no conflict of interest in this publication.

Acknowledgement
This research was partially supported by the JSPS KAKENHI (C), Grant No. 19K04144.

References

Buchholz, P., Kriege, J., & Felko, I. (2014). *Input modeling with phase-type distributions and Markov models: theory and applications*. Springer, Cham. ISBN 978-3-319-06674-5.

Inoue, S., & Yamada, S. (2006). Discrete software reliability assessment with discretized NHPP model. *Computers & Mathematics with Applications: An International Journal*, *51*(2), 161-170.

Inoue, S., & Yamada, S. (2007). Generalized discrete software reliability modeling with effect of program size. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *37*(2), 170-179.

Inoue, S., & Yamada, S. (2015). Software reliability assessment with multiple changes of testing-environment. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *98*(10), 2031-2041.

Langberg, N., & Singpurwalla, N.D. (1985). A unification of some software reliability model. *SIAM Journal on Scientific and Statistical Computing*, *6*(3), 781-790.

Musa, J.D., Iannino, A., & Okumoto, K. (1987). *Software reliability: measurement, prediction, application*. McGrow-Hill, New York.

Okamura, H., & Dohi, T. (2016). Phase type software reliability model: parameter estimation algorithms with grouped data. *Annals of Operations Research*, *244*(1), 177-208.

Yamada, S. (2014). *Software reliability modeling – fundamentals and applications* (Vol 5), Springer, Japan, Tokyo.