1998年度日本オペレーションズ・リサーチ学会
春季研究発表会

Software Safety/Reliability Modeling with Imperfect Debugging

01307475 鳥取大学 *得能貢一 TOKUNO Koichi
01702425 鳥取大学 山田茂 YAMADA Shigeru

1 Introduction

We develop a software safety/reliability assessment model which assumes that the system causes hazardous conditions randomly in operation. We use a Markov process to describe the time-dependent behavior of the software system, taking account of the software reliability growth process. Several quantitative safety/reliability measures are derived from this model. Especially, this model can provide a metric of *software safety* defined as the probability that the system does not fall into hazardous states at a specified time point [1]. Numerical illustrations are presented to show that this model is useful for software safety/reliability measurement and assessment.

2 Model description

We give the following assumptions to construct the software safety/reliability model in dynamic environment, taking account of the software failure-occurrence phenomenon:

A1. When the software system operates without software failure-occurrences, the holding times of the safe and unsafe state are distributed exponentially with means $1/\theta$ and $1/\eta$, respectively.

A2. A debugging activity is performed when a software failure occurs. Debugging activities are perfect with probability a $(0 \leq a \leq 1)$, while imperfect with probability $b(=1-a)$. We call a the perfect debugging rate.

A3. Software reliability growth occurs in case of the perfect debugging activity. The time-interval between software failure-occurrences is distributed exponentially with mean $1/\lambda_n$, where $n = 0, 1, 2, \ldots$ denotes the cumulative number of corrected faults.

A4. Only one fault is corrected and removed from the system in the state of perfect debugging activity and the debugging time is not considered.

The state space of stochastic process $\{X(t),\ t \geq 0\}$, which represents the state of the software system at time point t, is defined as follows:

W_n: the system is operating safely,

U_n: the system falls into the unsafe state.

From assumption A2, when the next software failure occurs in $\{X(t) = W_n\}$ or $\{X(t) = U_n\}$,

$$X(t) = \begin{cases} W_n & \text{(with probability } b) \\ W_{n+1} & \text{(with probability } a). \end{cases} \tag{1}$$

Further, we use Moranda model [2] to describe the software reliability growth process. That is, when n faults have been corrected, the hazard rate for the next software failure-occurrence, λ_n, is given by

$$\lambda_n = Dk^n \ (n = 0, 1, 2, \ldots;\ D > 0,\ 0 < k < 1), \tag{2}$$

where D and k are the initial hazard rate and the decreasing ratio of the hazard rate, respectively.

The sample state transition diagram of $X(t)$ is illustrated in Fig.1.

3 Software safety/reliability measures

The distribution of random variable S_n, which represents the time spent in correcting n faults, is obtained as

$$\begin{aligned} G_n(t) &\equiv \Pr\{S_n \leq t\} \\ &= \sum_{i=0}^{n-1} A_i^n \left[1 - e^{-a\lambda_i t}\right] \\ &\quad (t \geq 0;\ n = 1, 2, \ldots;\ G_0(t) \equiv 1), \end{aligned} \tag{3}$$

where constant coefficients A_i^n's are given by

$$\left.\begin{aligned} &A_0^1 \equiv 1 \\ &A_i^n = \prod_{\substack{j=0 \\ j \neq i}}^{n-1} \frac{\lambda_j}{\lambda_j - \lambda_i} \\ &(n = 2, 3, \ldots;\ i = 0, 1, 2, \ldots, n-1) \end{aligned}\right\}. \tag{4}$$

Further, the state occupancy probability that $X(t)$ is in state W_n at time point t is obtained as

$$\begin{aligned} P_{W_n}(t) &\equiv \Pr\{X(t) = W_n\} \\ &= B^n e^{-(\lambda_n+\theta+\eta)t} + \sum_{i=0}^{n} B_i^n e^{-a\lambda_i t} \\ &\quad (n = 0, 1, 2, \ldots), \end{aligned} \tag{5}$$

where constant coefficients B^n and B_i^n are given by

$$B^n = \frac{-\theta \prod_{j=0}^{n-1} a\lambda_j}{\prod_{j=0}^{n} (a\lambda_j - \lambda_n - \theta - \eta)}, \tag{6}$$

$$B_i^n = \frac{(\lambda_n + \eta - a\lambda_i) \prod_{j=0}^{n-1} \lambda_j}{(\lambda_n + \theta + \eta - a\lambda_i) \prod_{\substack{j=0 \\ j\neq i}}^{n} (\lambda_j - \lambda_i)}$$

$$(i = 0,\ 1,\ 2,\ \ldots,\ n), \tag{7}$$

respectively.

Then, *software safety* [3] is defined as

$$S(t) \equiv \sum_{n=0}^{\infty} P_{W_n}(t), \tag{8}$$

which represents the probability that the system does not fall into any unsafe states at time point t.

4 Numerical Examples

The software safety metrics, $S(t)$ in (8) for various values of θ are shown in Fig.2, where $D = 0.1$, $k = 0.8$, $a = 0.9$, and $\eta = 0.1$. Fig.2 indicates that the software safety becomes larger as θ decreases and converges to $\eta/(\theta+\eta)$, which denotes the steady probability that the system is operating safely in the case where software failure-occurrences are not considered.

$S(t)$'s are shown in Fig.3 for various values of k, where $D = 0.1$, $a = 0.9$, $\theta = 0.01$, $\eta = 0.1$. Fig.3 indicates that the software safety converges earlier with decreasing k. Smaller k means that software reliability growth occurs more rapidly. Since this model assumes that the system is not unsafe in causing a software failure, the software safety becomes larger with increasing k, which means the high frequency of software failure-occurrences.

References

[1] S.J. Keene, Jr., "Assuring software safety", Proc. Annu. Reliability and Maintainability Symp., Las Vegas, U.S.A., 1992, pp 274–279.

[2] P.B. Moranda, "Event-altered rate models for general reliability analysis", IEEE Trans. Reliability, vol R-28, no 5, 1979, pp 376–381.

[3] S. Yamada, K. Tokuno, Y. Kasano, "Quantitative assessment models for software safety/reliability" (in Japanese), Trans. IEICE A, vol J80-A, no 12, 1997, pp 2127–2137.

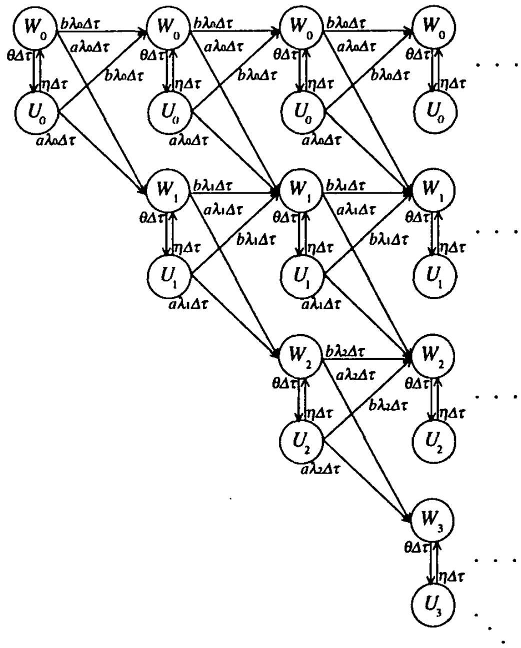

Fig.1 A diagrammatic representation of state transitions between $X(t)$'s.

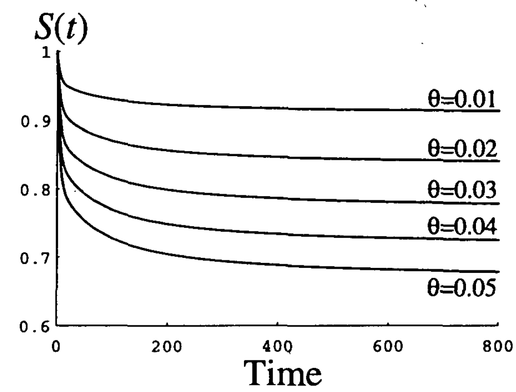

Fig.2 Dependence of θ on $S(t)$ ($D = 0.1$, $k = 0.8$, $a = 0.9$, $\eta = 0.1$).

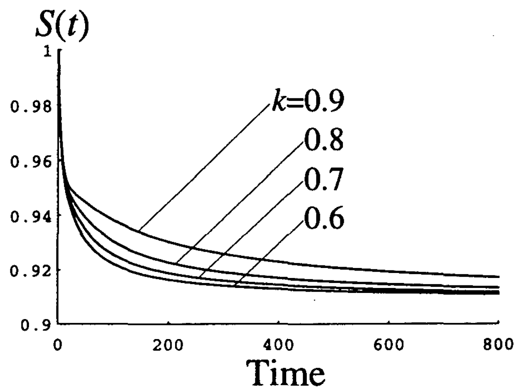

Fig.3 Dependence of k on $S(t)$ ($D = 0.1$, $a = 0.9$, $\theta = 0.01$, $\eta = 0.1$).