2001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

A Component-Based Jelinski & Moranda Software Reliability Model

H. Okamura† (01013754), S. Kuroki†, T. Dohi† (01307065), S. Osaki† (01002265)
†Graduate School of Engineering, Hiroshima University

1. Introduction

Software reliability is one of the most important factor in the development of modern computer systems, because the software errors of computer systems, which are utilized in financial institutions and other industries, will cause a great deal of damage to our lives. If we can develop a fault-free program, the software system will seldom fail. However, it will be impossible to detect all the software faults within a limited testing period and resources, especially for a complex and large scaled software. Hence, the software reliability has to be assessed quantitatively before being released to the user or market. Usually, the software reliability is estimated based on the stochastic dynamics of detecting software faults, *i.e.* a sequence of inter-failure time and cumulative number of faults. Then, the stochastic models called *software reliability growth models* are useful to assess the software reliability [1].

In general, the software reliability growth models can be classified into two types; a *black-box model* and a *white-box model*. The black-box model focuses only on the time series behavior of detected faults, such as a sequence of inter-failure time and cumulative number of faults. The most classical but well-known black-box model is the Jelinski & Moranda model [2]. In the Jelinski & Moranda model, it is assumed that the detection rate of software faults is proportional to the number of faults remaining in the software. More precisely, when the total number of faults in the software system, N (> 0), is given, it is assumed in the Jelinski & Moranda model that the fault detection rate depends on the cumulative number of faults experienced before, n ($< N$), and that the total number of detected faults follows the pure birth process with transition rate:

$$\mu_n = (N - n + 1)\mu, \tag{1}$$

where μ (> 0) is a positive constant.

However, it is pointed out that the black-box model needs obviously a large number of data to perform the accurate prediction, and ignores completely the software architecture which is the most important factor to affect the software reliability. Generally speaking, a software system is consisted of a number of modules or components, which can be regarded as functions in program sources. Since the software is executed in the unit of component on programmed logic, the program behavior should be considered to be stochastic in terms of execution time. This tells us that the software reliability strongly depends on both the component reliability for each active component and the transition behavior between components. From these reasons, the white-box models depending on the software structure recently receive considerable attentions to assess the reliability for highly critical software systems.

First, Littlewood [3, 4] proposed a component-based software reliability model. In the Littlewood model, it is assumed that the software component is executed according to a continuous-time Markov chain (CTMC). Since the seminal contribution by Littlewood [3, 4], many kinds of component-based software reliability models have been developed in the literature. Ledoux [5] developed an extended component-based model by introducing the concept of primary failures and secondary failures. Note that the Ledoux model is described as a versatile Markov process which is the most wide class of Markov point process. In this paper, we extend the classical Littlewood model from the different point of view. In the earlier literature [3-6], the component- based software reliability models are modeled to assess the software reliability in operational phase. Since the software test is sufficiently executed before releasing, it is assumed that the software system in the operational phase does not show the reliability growth phenomenon any longer. In other words, the existing component-based models focus on the operational reliability and do not take account of the reliability assessment in the testing. In the following section, we extend the Jelinski & Moranda model in terms of software structure and propose a somewhat different component-based software reliability model from Littlewood [3, 4].

2. Model Description

Let us consider a software system which consists of m components. One component is always being executed while the system is active. The currently executed component is called the *active component*. The active component frequently changes with the passage of time. It is assumed that the dynamic behavior of the active component is described by a CTMC with an infinitesimal generator $\boldsymbol{M}$. The active component at the initial time (initial

configuration) is selected based on the probability vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$. Software failures are caused only by faults in an active component, not in the other inactive ones. The fault detection rate depends on the components, where the fault detection rate vector, whose element corresponds to each component, is $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$. When a software failure caused by a fault occurs, then it is detected immediately, removed and/or repaired with no delayed time. Hence, after the repair, the software reliability will grow up.

In this paper, we make the following assumptions;

A. The interface failures at transition of an active component do not occur with probability one. Namely, there is no software fault in the interface among components.

B. The repair/remove time for a detected fault can be negligible.

C. The software system is reinitialized after repairing/removing a fault, *i.e.*, the software test is restarted from the initial software component configuration.

We define the size vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$ whose i-th element represents the relative size of component i, where the total sum of elements is 1. For example, when the software components can be regarded as computer programs, the component size can be interpreted as the amount of their source code. The number of faults in each component is assumed to be proportional to the size of component. Let N denote the number of faults in the software system at the initial time $t = 0$. Thus, the number of faults in each component can be expressed by $N\boldsymbol{\omega}$ at the initial time.

Let $P_{ij}(n,t)$ denote the probability that n faults are detected at time t and the active component changes from i to j, *i.e.*,

$$P_{ij}(n,t) = \Pr\{N(t) = n, J(t) = j \mid N(0) = 0, J(0) = i\}, \tag{2}$$

where $\{N(t),\ t \geq 0\}$ is the cumulative number of faults detected up to time t and $\{J(t),\ t \geq 0\}$ is the index on the active component. We define a matrix $\boldsymbol{P}(n,t)$ with elements $P_{ij}(n,t)$. From the well-known Chapman-Kolmogorov forward equation, we have

$$\frac{d}{dt}\boldsymbol{P}(0,t) = \boldsymbol{P}(0,t)\boldsymbol{M}(0), \tag{3}$$

$$\frac{d}{dt}\boldsymbol{P}(n,t) = \boldsymbol{P}(n,t)\boldsymbol{M}(n) - \boldsymbol{P}(n-1,t)\boldsymbol{M}(n-1)\boldsymbol{e}^T\boldsymbol{\alpha}, \quad (n = 1, 2, \cdots, N-1), \tag{4}$$

$$\frac{d}{dt}\boldsymbol{P}(N,t) = -\boldsymbol{P}(N-1,t)\boldsymbol{M}(N-1)\boldsymbol{e}^T\boldsymbol{\alpha}, \tag{5}$$

where $\boldsymbol{e}^T$ is a column vector with elements 1, and

$$\boldsymbol{M}(n) = \boldsymbol{M} - (N-n)\mathrm{diag}(\omega_1\mu_1, \ldots, \omega_m\mu_m). \tag{6}$$

If the software involves only one component, the present model is then reduced to the Jelinski & Moranda model [2]. In other words, our model includes the Jelinski & Moranda model as a special case, and can be regarded as an extended model taking account of the software architecture.

From the familiar Markovian argument, it is easily seen that the MTBF (mean time between failures) for the n-th detected fault is

$$\mathrm{MTBF}(n) = \boldsymbol{\alpha}\{\boldsymbol{M}(n-1)\}^{-1}\boldsymbol{e}^T. \tag{7}$$

By solving the difference-differential equations (4)–(6), the expected cumulative number of faults detected up to time t is given by

$$\mathrm{E}[N(t)] = \boldsymbol{\alpha}\sum_{n=0}^{N} n\boldsymbol{P}(n,t)\boldsymbol{e}^T. \tag{8}$$

Further differentiating $\mathrm{E}[N(t)]$ with respect of t yields

$$\frac{d}{dt}\mathrm{E}[N(t)] = -\boldsymbol{\alpha}\sum_{n=0}^{N-1}\boldsymbol{P}(n,t)\boldsymbol{M}(n)\boldsymbol{e}^T. \tag{9}$$

Hence, the instantaneous MTBF and the cumulative MTBF are given by

$$\mathrm{MTBF}_I(t) = \frac{1}{d\mathrm{E}[N(t)]/dt} \tag{10}$$

and

$$\mathrm{MTBF}_C(t) = \frac{t}{\mathrm{E}[N(t)]}, \tag{11}$$

respectively.

References

[1] M. R. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, NY, 1996.

[2] Z. Jelinski and P. B. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation* (W. Freiberger ed.), pp. 465–484, 1979.

[3] B. Littlewood, "A reliability model for systems with Markov structure," *Applied Statics.*, vol. 24, pp. 172–177, 1975.

[4] B. Littlewood, "Software reliability model for modular program structure," *IEEE Trans. Reliability*, vol. 28, pp. 241–246, 1980.

[5] J. Ledoux, "Availability modeling of modular software," *IEEE Trans. Reliability*, vol. 48, pp. 159–168, 1999.