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

Preventive Maintenance for a Software System with High Assurance Requirement

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

1. Introduction

The software rejuvenation is one of the most effective preventive maintenance technique for operational software systems with high assurance requirement. Garg, *et al.* [1] and Okamura, *et al.* [2] proposed two types of periodic rejuvenation policies for a transaction based software system, where transactions arrive at the system and are processd randomly. In this article, we develop new software rejuvenation policies based on the number of transactions completed. Applying the familiar technique based on the hidden Markov process, we give numerical computation procedures on the system dependability measures, such as steady-state availability and the loss probability of transactions.

2. Model Description

Consider the similar but somewhat different software system with single server from Garg, *et al.* [1]. Suppose that a buffer is empty at time $t = 0$ and that the transactions arrive at the system in accordance with the homogeneous Poisson process with rate λ (> 0). If the buffer is empty, then the process for the transactions are started with service rate $\mu(\cdot)$ (> 0), otherwise, the transactions are accumulated in the buffer until the server is available, where the service rate $\mu(\cdot)$ is a function dependent of the cumulative operation time t and/or the number of transactions in the buffer. More precisely, let $\{X_t; t \geq 0\}$ be the number of transactions accumulated in the buffer at time t. Then, the service rate is given by $\mu(t)$, $\mu(X_t)$ or $\mu(t, X_t)$. It is assumed that the buffer size is fixed as K (> 1). If a transaction arrives at the system when the number of transactions in the buffer is K, it will be rejected from the system. We also suppose that the server is unreliable and that the failure rate $\rho(\cdot)$ also depends on the cumulative operation time t and/or the number of transactions in the buffer. If the system fails, then the recovery (repair) is started immediately, where the recovery time is the i.i.d. random variable Y_r with $\mathrm{E}[Y_r] = \gamma_r$ (> 0). If an additional transaction arrives at the system during the recovery operation, it will be lost.

On the other hand, the software rejuvenation is motivated if the software failure rate $\rho(\cdot)$ is increasing in t. This IFR (Increasing Failure Rate) assumption can be validated from the phenomenon called *the software aging* observed in actual operating systems or midware systems. Consider the similar but somewhat different software rejuvenation policies from Garg, *et al.* [1]:

Policy I: Rejuvenate the system at the time when N (> 1) transactions are processed. If one or more transactions are remained at that time in the buffer, they will be all lost from the buffer.

Policy II: Rejuvenate the system at the beginning of the idle period after the system completed N (> 1) transactions.

Only difference from Garg, *et al.* [1] is to start the software rejuvenation preventively at the random time when the total number of transactions completed to process reaches to a threshold level N. Let Y_R be the preventive maintenance time related with the rejuvenation and be the i.i.d. random variable with $\mathrm{E}[Y_R] = \gamma_R$ (> 0). After completing the rejuvenation, the system is restarted with empty buffer.

3. Dependability Measures

Consider three dependability measures; steady-state availability, loss probability of transactions and mean response time on transactions. We also apply the similar method based on the hidden Markov process to Garg, *et al.* [1]. Consider a discrete Markov chain with three states, where

State A: system operation

State B: recovery from software failure

State C: preventive maintenance (rejuvenation).

Define the transition probabilities P_{AB} and P_{AC} from state A to state B and from state A to state C, where $P_{AC} = 1 - P_{AB}$. Then, since the transition probability matrics is

$$P = \begin{bmatrix} 0 & P_{AB} & P_{AC} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

we obtain the steady-state probabilities in respective states as $\pi_A = 0.5$, $\pi_B = 0.5P_{AB}$ and $\pi_C = 0.5P_{AC}$. Also, define the following random variables:

U: sojourn time in state A in the steady state

U_n: sojourn time in state A when n ($n = 0, \cdots, K$) transactions are stored in the buffer ($U = \sum_{n=0}^{K} U_n$)

N_l: number of transactions lost in the transition from state A to state B or C

(i) Steady state availability:

$$A_{SS} = \frac{\mathrm{E}[U]}{\mathrm{E}[U] + P_{AB}\gamma_r + P_{AC}\gamma_R}. \tag{1}$$

(ii) Loss probability of transactions:

$$P_{loss} = \frac{\lambda(P_{AB}\gamma_r + P_{AC}\gamma_R + \mathrm{E}[U_K]) + \mathrm{E}[N_l]}{\lambda(\mathrm{E}[U] + P_{AB}\gamma_r + P_{AC}\gamma_R)}. \tag{2}$$

(iii) Mean response time on transactions:

Let E and W_s be the mean number of transactions serviced from the system and the corresponding expected total processing time, respectively. Then, we obtain $E = \lambda(\mathrm{E}[U] - \mathrm{E}[U_K])$. From an intuitive argument, the expected total processing time for all transactions arrived at the system is $W = \sum_{n=0}^{K} n\mathrm{E}[U_n]$. Since $W_s < W$, the mean response time on transactions becomes

$$T_{res} = \frac{W_s}{E - \mathrm{E}[N_l]}. \tag{3}$$

From the inequality $W_s < W$, we obtain an upper bound of the mean response time as follows.

$$T_{res} < \frac{W}{E - \mathrm{E}[N_l]}. \tag{4}$$

4. Analysis

Of our interest is the derivation of the optimal threshold level N^* under Policy I and Policy II. Hence, we define the steady-state availability and the loss probability of transactions as the functions of N again, *i.e.* $A_{SS}(N)$ and $P_{loss}(N)$. To seek P_{AB}, $\mathrm{E}[U_n]$ and $\mathrm{E}[N_l]$ in these dependability measures, define the number of transactions completed to process until time t, the probability that n transactions are remained in the buffer at time t and the probability that i transactions are completed to process provided that n transactions are remained in the buffer at time t, by $\{Z_t;\ t \geq 0\}$, $q_n(t)$ and $q_n^{(i)}(t)$ ($i = 1, \cdots, N; n = 0, \cdots, K$), respectively. Since

$$q_n(t) = \Pr\{X_t = n\}, \tag{5}$$

$$q_n^{(i)}(t) = \Pr\{Z_t = i | X_t = n\}, \quad (i = 1, \cdots, N; n = 0, \cdots, K), \tag{6}$$

the joint probability of X_t and Z_t is represented by

$$p_n^{(i)}(t) = \Pr\{N_t = n, Z_t = i\} = q_n(t) \cdot q_n^{(i)}(t). \tag{7}$$

Under Policy I, we consider a continuous-time Markov chain (CTMC) with states

$0^{(i)}, \cdots, K^{(i)}$: i ($i = 0, \cdots, N-1$) transactions are completed, where $0 \sim K$ means the number of transactions in the buffer,

$0^{(N)}, \cdots, K^{(N)}$: after N transactions are completed to process, the system is started to rejuvenate (absorbing states),

$0', \cdots, K'$: the software failure occurs (absorbing states).

Applying the well-known state-space method, we can formulate the Kolmogorov's forward equation which is the difference-differential equations on $p_n^{(i)}(t)$ and $p_{n'}(t)$. Finally, under Policy I, we derive

$$P_{AB} = \sum_{n=0}^{K} p_{n'}(\infty), \tag{8}$$

$$\mathrm{E}[U_n] = \int_0^\infty \sum_{i=0}^{N-1} p_n^{(i)}(t)dt, \tag{9}$$

$$\mathrm{E}[N_l] = \sum_{n=0}^{K} n(p_{n'}(\infty) + p_n^{(N)}(\infty)). \tag{10}$$

In a fashion similar to Policy I, we get the results for Policy II as follows.

$$P_{AB} = \sum_{n=0}^{K} p_{n'}(\infty), \tag{11}$$

$$\mathrm{E}[U_0] = \int_0^\infty \sum_{i=0}^{N-1} p_0^{(i)}(t)dt, \tag{12}$$

$$\mathrm{E}[U_n] = \int_0^\infty \sum_{i=0}^{N} p_n^{(i)}(t)dt, \quad (n = 1, \cdots, K) \tag{13}$$

$$\mathrm{E}[U_l] = \sum_{n=0}^{K} np_{n'}(\infty). \tag{14}$$

References

[1] S. Garg, A. Puliafito, M. Telek and K. Trivedi, Analysis of preventive maintenance in transactions based software systems, *IEEE Trans. on Computers*, **47**, 96–107 (1998).

[2] H. Okamura, A. Fujimoto, T. Dohi, S. Osaki and K. S. Trivedi, The optimal preventive maintenancc policy for a software system with multi server station, *Proc. 6th ISSAT Int'l. Conf. on Reliability and Quality in Design*, 275–279 (2000).