RAIRO Operations Research
RAIRO Oper. Res. **36** (2002) 129-148
DOI: 10.1051/ro:2002010

REFINED NON-HOMOGENEOUS MARKOVIAN MODELS FOR A SINGLE-SERVER TYPE OF SOFTWARE SYSTEM WITH REJUVENATION

Hiroyuki Okamura[1], S. Miyahara[1] and T. Dohi[1]

Communicated by Naoto Kaio

Abstract. Long running software systems are known to experience an aging phenomenon called software aging, one in which the accumulation of errors during the execution of software leads to performance degradation and eventually results in failure. To counteract this phenomenon a proactive fault management approach, called software rejuvenation, is particularly useful. It essentially involves gracefully terminating an application or a system and restarting it in a clean internal state. In this paper, we reconsider the non-homogeneous Markovian models for a single-server type of software system with rejuvenation in Garg *et al.* (1998), and revisit them from the theoretical view point. More precisely, it is assumed in these models that software failures can occur with positive probability during idle periods in transaction systems, but we exclude this unreasonable situation in our refined models.

Keywords: Preventive maintenance, aging, rejuvenation, software system, dependability, Markovian analysis, queue.

Received June, 2002.

[1] Department of Information Engineering, Graduate School of Engineering, Hiroshima University, 1–4–1 Kagamiyama, Higashi-Hiroshima 739–8527, Japan;
e-mail: okamu@rel.hiroshima-u.ac.jp

1. Introduction

It has been recognized for a long time that the software system does not deteriorate in the operational phase. However, the phenomenon called *software aging* is observed by some authors [1, 13], especially in operating systems and widely-used applications like Netscape and Internet Explorer. The software aging will affect the performance of the application and eventually cause it to fail. Huang *et al.* [10] report this phenomenon in telecommunications billing applications where over time the application experiences a crash or a hang failure. Avritzer and Weyuker [2] discuss aging in a telecommunication switching software where the effect manifests as gradual performance degradation. Of course, the software aging phenomenon is caused by some software faults which induce the performance degradation such as the memory leak and the fragmentation. Hence, software users may avoid the degradation if they can detect and fix the faults. However, these works are not always possible because the application developer prohibits accessing to the source code directly in many cases. Furthermore, even if the accessing was permitted, it would be almost impossible for software users to detect the faults causing the degradation in the software in which various programs are entangled intricately. In addition, common experience suggests that most software failures caused by the software aging are transient in nature [7]. Since the transient failure will disappear if the same operation as one at the failure occurrence is retried later in slightly different context, it is also difficult to detect their roots. Therefore, the software aging phenomenon and the transient software failure have to be tolerated in the operational phase. Usual strategies to deal with the transient failure in the operational phase are passive in nature; they consist of action taken after failure.

Recently, a complementary approach to handle transient software failures, called *software rejuvenation*, is proposed [10]. Software rejuvenation is a preventive solution that is particularly useful for counteracting the phenomenon of software aging. It involves stopping the running software occasionally, cleaning its internal state and restarting it. Cleaning the internal state of a software might involve *garbage collection*, *flushing operating system kernel tables*, *reinitializing internal data structures*, etc. An extreme, but well-known example of rejuvenation is a *hardware reboot*. Apart from being used in an ad-hoc manner by almost all software users, the software rejuvenation has been used in high availability and mission critical systems [14, 15]. Although the fault in the program still remains, performing the rejuvenation occasionally or periodically prevents severe failures due to that fault.

Rejuvenation has the same motivation, advantages and disadvantages as preventive maintenance policies in hardware systems. Any rejuvenation typically involves an overhead, but, on the other hand, prevents more severe failures to occur. The application will of course be unavailable during rejuvenation. However, since this is a scheduled downtime, the cost is expected to be much lower than the cost of an unscheduled downtime caused by failure. Hence, an important issue is to determine the optimal schedule to perform software rejuvenation in terms of dependability measures.

Huang *et al.* [10] analyze a continuous-time Markov chain model with software rejuvenation. Garg *et al.* [4] introduce the idea of periodic rejuvenation and extend the original Huang *et al.* model [10]. To deal with deterministic interval between successive rejuvenations the system behavior was represented through a Markov regenerative stochastic Petri net. Dohi *et al.* [3] develop semi-Markov models to estimate the cost-optimal rejuvenation schedule from the empirical failure data. As an alternative modeling approach, the work in Garg *et al.* [5] involves arrival and queueing of transactions or jobs in the system, and computes load and time dependent rejuvenation policy. The above models consider the effect of aging as crash/hang failure, referred to as hard failures, which result in unavailability of the software. However, due to the aging the software system can exhibit soft failures, that is, performance degradation. Both effects of aging, hard failures that result in an unavailability and soft failures that result in performance degradation, are considered in the model of transaction based software system. In particular, Garg *et al.* [6] propose the time based software rejuvenation scheme for a server type of software system, where the rejuvenation starts at any timing measured by the cumulative operation time. Recently, an alternative control policy, the workload based policy, is proposed in the same modeling framework [12].

In this paper, we revisit the seminal software rejuvenation schemes by Garg *et al.* [6] and Okamura *et al.* [12] from the theoretical view point. Garg *et al.* [6] consider the time based rejuvenation policy for the typical transaction based software system, but deal with only the case where the server may fail even during idle periods. As expected easily, most types of software systems can fail in the operative time, but seldom does during idle periods. In other words, the model by Garg *et al.* [6] fails to describe such a delicate behavior of the transaction based software system. This paper reformulates the single-server type of software system and derive the dependability measures under the plausible assumption that the server cannot fail during idle periods. Further, we also deal with the alternative control policy for the software rejuvenation, which is not based on the cumulative operation time but on the number of completed transactions. This new control scheme is proposed by the authors [12]. The difference between these two models is quite similar to that between *T-policy* and *N-policy* in the classical queueing theory [8, 9]. Finally, numerical examples are presented to quantitatively examine the effect on the failure during idle periods.

2. Software rejuvenation scheme

Let us consider a software rejuvenation scheme for a single-server type of software system. The original model is proposed by Garg *et al.* [6]. The arrival stream of transactions follows the homogeneous Poisson process with parameter λ (>0). The transaction is served at the rate of $\mu(\cdot)$ (>0) per unit time and under the first-come first-served discipline. The server has a buffer, whose capacity is K (>1). The transactions while the server processes another transactions can be always stored in the buffer, but, if the number of transactions in the buffer is larger than

the capacity, the transactions that arrive later may be lost. The server exhaustively processes the transactions while at least one transaction is in the buffer. By contrast, only when the buffer becomes empty, the server may be idle and wait for a new transaction. The time period is called an *idle period*.

The server may fail at the rate of $\rho(\cdot)$ (>0) per unit time. The failure is caused by the software aging phenomenon such as memory leaking, data corruption and fragmentation. Taking account of the software aging, we assume that the failure rate is an increasing function of time. Note that, in our modeling framework, the service rate deceases and the failure rate increases as the operation time elapses. After the server fails, the system starts the recovery operation where the recovery time (recovery overhead) is a non-negative random variable Y_r with $\mathrm{E}[Y_r] = \gamma_r$ (>0). Before starting the recovery operation, all the transactions stored in the buffer may be lost. In addition, all the transactions arriving at the system during the recovery operation may be also lost.

The deterioration of the system performance, increase of the failure rate and decrease of the service rate, motivates the software rejuvenation [6, 10]. In this paper, the software rejuvenation is executed under the following policies [6, 12]:

Policy I [6]: Rejuvenate the system when the operation time becomes the threshold time T (>0). The system is forced to execute the software rejuvenation at that time, even if there exist transactions in the buffer. Thus all the transactions stored in the buffer will be lost.

Policy II [6]: Rejuvenate the system after the operation time becomes T (>0) but the software rejuvenation is executed only when the buffer becomes empty.

Policy III [12]: Rejuvenate the system when the number of completed transactions becomes the threshold level N (>1). The software rejuvenation is executed immediately. Then all the transactions stored in the buffer will be lost.

Policy IV [12]: Rejuvenate the system after the number of completed transactions is N (>1), but the software rejuvenation is executed only when the buffer is empty.

Let Y_R be the preventive maintenance time to rejuvenate the system (rejuvenation overhead) with mean $\mathrm{E}[Y_R] = \gamma_R$ (>0). In the software rejuvenation as well as the recovery operation, all the transactions arriving at the system may be lost.

Figures 1 and 2 depict the possible behavior of the single-server type of software system with rejuvenation. Since Policies I and II are essentially based on the operation time, both of them are called *T-policy*. On the other hand, since Policies III and IV depend on the number of completed transactions, we thus call them *N-policy*. The essential difference between T and N policies is the timing to execute the software rejuvenation. The rejuvenation timings under Policies I and II are deterministic, by contrast, the rejuvenation timings under Policies III and IV are random.

In the above modeling framework with four kinds of rejuvenation policies, we make a plausible assumption that the server never fails during the idle period. In

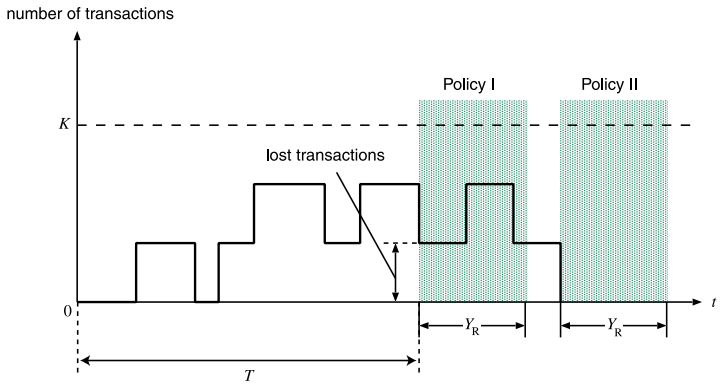

FIGURE 1. Possible behavior of a single-server type of software system with rejuvenation under Policies I and II.

the same framework, Garg *et al.* [6] and Okamura *et al.* [12] make the assumption that the server can fail during the idle period, but this is a questionable assumption. Strictly speaking, the server has nothing to do in the idle period. In the idle period, the important task of the server is to monitor arrivals of requests. It does not usually cause the system failure. Therefore the assumption that the server never fails during the idle period will be validated.

3. Dependability measures

Consider three dependability measures; steady-state availability, probability of transaction loss and mean response time on transactions. The behavior of transaction system can be modeled by the discrete time Markov chain (DTMC) with three states (see Fig. 3).

State A: available state;
State B: failure/recovery state;
State C: rejuvenation state.

With the above states, we consider a coupled stochastic process $\{(X_t, S_t);\ t \geq 0\}$ consisting of both the number of transactions stored in the buffer and the state of the software system. The decomposed stochastic process $\{X_t;\ t \geq 0\}$ is described by a non-homogeneous Markov process (NHMP). However, the Markovian property does not hold for the other process $\{S_t;\ t \geq 0\}$, since the recovery and the rejuvenation overheads are generally distributed random variables. In fact, the coupled stochastic process (X_t, S_t) is the Markov regenerative process (MRGP) [4, 11]. The difference on the rejuvenation polices and the model assumption will influence this coupled stochastic process.

Our analysis for the dependability measures contains two steps. In the first step, by using the hidden Markovian analysis, the dependability measures are formulated with probabilistic quantities, transition probability from State A to

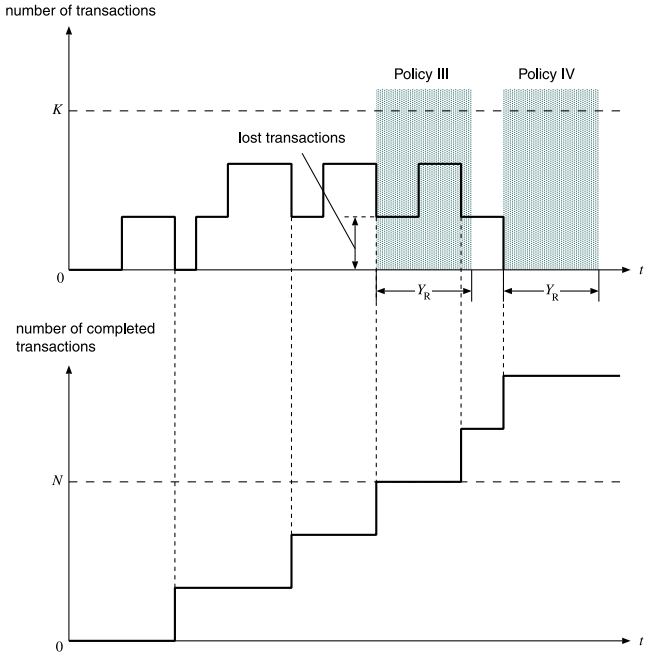

FIGURE 2. Possible behavior of a single-server type of software system with rejuvenation under Policies III and IV.

State B, expected sojourn time in State A and expected number of transaction loss at the time points of state changes. In the second step, the probabilistic quantities above are calculated by using the well-known Markovian state space method.

3.1. FORMULATION

In order to derive the dependability measures, we will apply the well-known hidden Markovian method. At the time points of state changes, the software system can be described as a DTMC with the following transition probability matrix,

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

where P_{AB} and P_{AC} denote the respective transition probabilities from State A to State B and State A to State C. The steady-state probabilities for the DTMC can be easily derived by

$$\pi_A = 1/2, \tag{1}$$

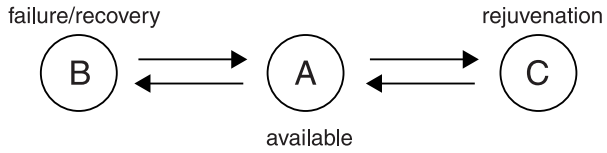

FIGURE 3. The transition diagram of the DTMC.

$$\pi_B = P_{AB}/2 \tag{2}$$

and

$$\pi_C = P_{AC}/2. \tag{3}$$

Define the following random variables in the steady state:

U**:** the length of sojourn time in State A;

U_n**:** the length of sojourn time in State A with n $(= 0, \ldots, K)$ transactions stored in the buffer, namely,

$$U = \sum_{n=0}^{K} U_n, \qquad \text{with probability one;} \tag{4}$$

N_l**:** the number of lost transactions at the time points of the state changes from State A to State B or State A to State C.

3.1.1. *Steady-state availability*

From the renewal reward argument, the steady-state availability is formulated as

$$A_{ss} = \frac{\pi_A \mathrm{E}[U]}{\pi_A \mathrm{E}[U] + \pi_B \gamma_r + \pi_C \gamma_R}\,. \tag{5}$$

Substituting equations (1–3) into equation (5), it can be seen that

$$A_{ss} = \frac{\mathrm{E}[U]}{\mathrm{E}[U] + P_{AB}\gamma_r + P_{AC}\gamma_R}\,. \tag{6}$$

3.1.2. *Probability of transaction loss*

The transaction loss occurs if

(a): transactions arrive at the system during the recovery operation;
(b): transactions arrive at the system during the software rejuvenation;
(c): transactions overflow the buffer;
(d): the server fails;
(e): the software rejuvenation is executed under Policies I and III.

In the cases (d) and (e), all the transactions stored in the buffer may be lost, and the expected number of lost transactions is expressed by $\mathrm{E}[N_l]$. Since the arrival stream follows the homogeneous Poisson process, the expected numbers of lost transactions in the cases (a) and (b) are given by $\lambda\gamma_r$ and $\lambda\gamma_R$, respectively. In the case (c), the expected number of lost transactions is $\lambda\mathrm{E}[U_K]$. Therefore, the probability of transaction loss is given by

$$P_{\mathrm{loss}} = \frac{\lambda(P_{AB}\gamma_r + P_{AC}\gamma_R + \mathrm{E}[U_K]) + \mathrm{E}[N_l]}{\lambda(\mathrm{E}[U] + P_{AB}\gamma_r + P_{AC}\gamma_R)}\,. \tag{7}$$

3.1.3. *Mean response time on transactions*

Let W_s be the expected total amount of response time for the completed transactions. When C denotes the expected number of completed transactions, it can be derived that $C = \lambda(\mathrm{E}[U] - \mathrm{E}[U_K])$. Thus the mean response time on transactions is given by

$$T_{\mathrm{res}} = \frac{W_s}{C - \mathrm{E}[N_l]}. \tag{8}$$

Here, since the expected total amount of response time for all transactions,

$$W = \sum_{n=0}^{K} n\mathrm{E}[U_n], \tag{9}$$

is always larger than W_s, an upper bound of the mean response time is given by

$$T_{\mathrm{res}} < \frac{W}{C - \mathrm{E}[N_l]} = \overline{T}_{\mathrm{res}}. \tag{10}$$

In this way, we formulate implicitly the steady-state availability, the probability of transaction loss and the upper bound of the mean response time on transactions, based on the simple DTMC. In the subsequent section, these dependability measures are represented as the functions of the decision variable, T or N.

3.2. Analysis for T-policy

Consider a continuous-time NHMP with the following states and corresponding probabilities:

State $0, \ldots, K$**:** the system is in State A, where $0, \ldots, K$ means the number of transactions in the buffer;

State $0', \ldots, K'$**:** the server fails and $0, \ldots, K$ transactions are lost at the failure point (absorbing states);

$p_n(t)$**:** probability that n transactions are in the buffer at time t;

$p_{n'}(t)$**:** probability that the server fails and that n transactions are lost at the failure.

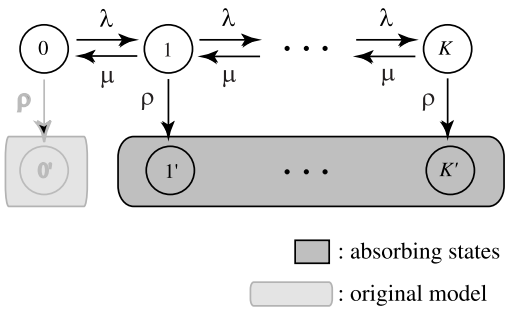

FIGURE 4. Modified state diagram under the Policy I.

3.2.1. *Policy I*

Applying the Markovian state space method to the NHMP, Garg *et al.* [6] formulate the difference-differential (Kolmogorov's forward) equations on $p_n(t)$ and $p_{n'}(t)$ as follows:

$$\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} = \mu(\cdot)p_1(t) - \{\lambda + \rho(\cdot)\}p_0(t), \tag{11}$$

$$\frac{\mathrm{d}p_n(t)}{\mathrm{d}t} = \mu(\cdot)p_{n+1}(t) + \lambda p_{n-1}(t) - \{\mu(\cdot) + \lambda + \rho(\cdot)\}p_n(t), \quad n = 1, \ldots, K-1, \tag{12}$$

$$\frac{\mathrm{d}p_K(t)}{\mathrm{d}t} = \lambda p_{K-1}(t) - \{\mu(\cdot) + \rho(\cdot)\}p_K(t), \tag{13}$$

$$\frac{\mathrm{d}p_{n'}(t)}{\mathrm{d}t} = \rho(\cdot)p_n(t), \quad n = 0, \cdots, K. \tag{14}$$

The formulation by Garg *et al.* [6] includes the case where the server fails during the idle period. Hence, we reformulate the difference-differential equations by reducing the failure rate in idle periods to zero. The resulting difference-differential equations are expressed in the following forms (see Fig. 4):

$$\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} = \mu(\cdot)p_1(t) - \lambda p_0(t), \tag{15}$$

$$\frac{\mathrm{d}p_n(t)}{\mathrm{d}t} = \mu(\cdot)p_{n+1}(t) + \lambda p_{n-1}(t) - \{\mu(\cdot) + \lambda + \rho(\cdot)\}p_n(t), \quad n = 1, \ldots, K-1, \tag{16}$$

$$\frac{\mathrm{d}p_K(t)}{\mathrm{d}t} = \lambda p_{K-1}(t) - \{\mu(\cdot) + \rho(\cdot)\}p_K(t), \tag{17}$$

$$\frac{\mathrm{d}p_{n'}(t)}{\mathrm{d}t} = \rho(\cdot)p_n(t), \quad n = 1, \cdots, K. \tag{18}$$

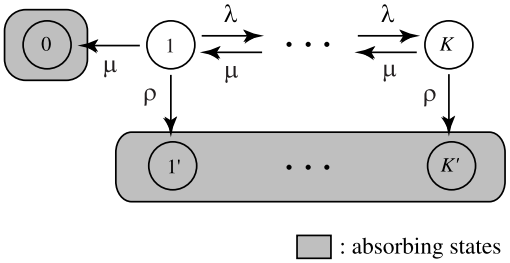

FIGURE 5. State diagram under Policy II ($t > T$).

Solving the difference-differential equations yields

$$P_{AB} = \sum_{n=1}^{K} p_{n'}(T), \tag{19}$$

$$E[U_n] = \int_{t=0}^{T} p_n(t)\mathrm{d}t, \quad n = 0, \ldots, K, \tag{20}$$

$$E[N_l] = \sum_{n=1}^{K} n \{p_n(T) + p_{n'}(T)\} \cdot \tag{21}$$

From equations (19–21), the dependability measures are calculated under Policy I.

3.2.2. *Policy II*

In a fashion similar to Policy I, the difference-differential equations on $p_n(t)$ and $p_{n'}(t)$ are derived under Policy II. In Policy II, the software rejuvenation is executed at the beginning of the first idle period after the operation time reaches the threshold time T. Thus the difference-differential equations for Policy II during $0 \le t \le T$ are equivalent to those for Policy I. On the other hand, the equations for $t > T$ can be obtained as follows (see Fig. 5).

$$\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} = \mu(\cdot)p_1(t), \tag{22}$$

$$\frac{\mathrm{d}p_1(t)}{\mathrm{d}t} = \mu(\cdot)p_2(t) - \{\mu(\cdot) + \lambda + \rho(\cdot)\}p_1(t), \tag{23}$$

$$\frac{\mathrm{d}p_n(t)}{\mathrm{d}t} = \mu(\cdot)p_{n+1}(t) + \lambda p_{n-1}(t) - \{\mu(\cdot) + \lambda + \rho(\cdot)\}p_n(t), \quad n = 2, \ldots, K-1, \tag{24}$$

$$\frac{\mathrm{d}p_K(t)}{\mathrm{d}t} = \lambda p_{K-1}(t) - \{\mu(\cdot) + \rho(\cdot)\}p_K(t), \tag{25}$$

$$\frac{\mathrm{d}p_{n'}(t)}{\mathrm{d}t} = \rho(\cdot)p_n(t), \quad n = 1, \ldots, K. \tag{26}$$

The above formula is the same as Garg *et al.* [6]. In fact, one does not need to reformulate the difference-differential equations under Policy II, because the system does not become idle without the software rejuvenation for $t > T$. Equations (22–26) lead to the following probabilistic quantities:

$$P_{AB} = \sum_{n=1}^{K} p_{n'}(\infty), \tag{27}$$

$$E[U_0] = \int_{t=0}^{T} p_0(t)\mathrm{d}t, \tag{28}$$

$$E[U_n] = \int_{t=0}^{\infty} p_n(t)\mathrm{d}t, \quad n = 1, \ldots, K, \tag{29}$$

$$E[N_l] = \sum_{n=1}^{K} n p_{n'}(\infty). \tag{30}$$

3.3. Analysis for N-policy

Consider a continuous-time NHMP with the following $(N+2) \times (K+1)$ states and their corresponding probabilities,

State $0^{(i)}, \ldots, K^{(i)}$**:** The system completes i $(= 0, \ldots, N-1)$ transactions when $0, \ldots, K$ transactions remain in the buffer;
State $0^{(N)}, \ldots, K^{(N)}$**:** The system completes N or more transactions when $0, \ldots, K$ transactions remain in the buffer;
State $0', \ldots, K'$**:** The system fails with $0, \ldots, K$ transactions stored in the buffer (absorbing states);
$p_n^{(i)}(t)$**:** probability that the system completes i transactions until time t and that n transactions remain in the buffer at time t;
$p_{n'}(t)$**:** probability that the system fails and loses n transactions until time t.

3.3.1. *Policy III*

Figure 6 illustrates the Markovian state diagram in the case of Policy III. Since the system executes the software rejuvenation just after N transactions are completed, the states $0^{(N)}, \ldots, K^{(N)}$ are the absorbing states and therefore the number of states indicates the number of lost transactions. The difference-differential equations on $p_n^{(i)}(t)$ and $p_{n'}(t)$ consist of four types of difference-differential equations in respective cases.

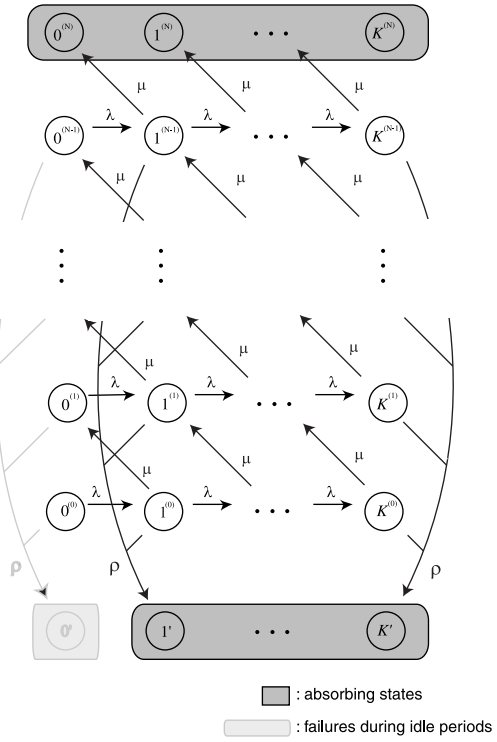

FIGURE 6. Modified state diagram under Policy III.

Case (i): no transaction is completed,

$$\frac{\mathrm{d}p_0^{(0)}(t)}{\mathrm{d}t} = -\lambda p_0^{(0)}(t), \tag{31}$$

$$\frac{\mathrm{d}p_n^{(0)}(t)}{\mathrm{d}t} = \lambda p_{n-1}^{(0)}(t) - \{\lambda + \mu(\cdot) + \rho(\cdot)\}p_n^{(0)}(t), \quad n = 1, \ldots, K-1, \tag{32}$$

$$\frac{\mathrm{d}p_K^{(0)}(t)}{\mathrm{d}t} = \lambda p_{K-1}^{(0)} - \{\mu(\cdot) + \rho(\cdot)\}p_K^{(0)}(t). \tag{33}$$

Case (ii): $1, \ldots, N-1$ transactions are completed,

$$\frac{\mathrm{d}p_0^{(i)}(t)}{\mathrm{d}t} = \mu(\cdot)p_1^{(i-1)}(t) - \lambda p_0^{(i)}(t), \tag{34}$$

$$\frac{\mathrm{d}p_n^{(i)}(t)}{\mathrm{d}t} = \lambda p_{n-1}^{(i)}(t) + \mu(\cdot)p_{n+1}^{(i-1)}(t) - \{\lambda + \mu(\cdot) + \rho(\cdot)\}p_n^{(i)}(t), \quad n = 1, \ldots, K-1, \tag{35}$$

$$\frac{\mathrm{d}p_K^{(i)}(t)}{\mathrm{d}t} = \lambda p_{K-1}^{(i)}(t) - \{\mu(\cdot) + \rho(\cdot)\}p_K^{(i)}(t). \tag{36}$$

Case (iii): the software rejuvenation is executed,

$$\frac{\mathrm{d}p_n^{(N)}(t)}{\mathrm{d}t} = \mu(\cdot)p_{n+1}^{(N-1)}(t), \qquad n = 0, \ldots, K-1. \tag{37}$$

Case (iv): the server fails,

$$\begin{aligned}\frac{\mathrm{d}p_{n'}(t)}{\mathrm{d}t} &= \rho(\cdot)p_n^{(0)}(t) + \cdots + \rho(\cdot)p_n^{(N-1)}(t) \\ &= \sum_{i=0}^{N-1} \rho(\cdot)p_n^{(i)}(t), \quad n = 1, \ldots, K.\end{aligned} \tag{38}$$

Note that the above difference-differential equations represent the case where the server never fail in the idle period.

By solving the difference-differential equations above, the probabilistic quantities are given by

$$P_{AB} = \sum_{n=1}^{K} p_{n'}(\infty), \tag{39}$$

$$\mathrm{E}[U_n] = \int_0^\infty \sum_{i=0}^{N-1} p_n^{(i)}(t)\mathrm{d}t, \quad n = 0, \ldots, K, \tag{40}$$

$$\mathrm{E}[N_l] = \sum_{n=1}^{K} n \left\{ p_{n'}(\infty) + p_n^{(N)}(\infty) \right\}. \tag{41}$$

The assumption that the server never fails in idle periods also gives an effect on the probabilistic quantities above. If the assumption was not made under Policy III, equation (39) should be expressed by the sum of $p_{n'}(\infty)$ from $n = 0$ to K.

3.3.2. *Policy IV*

Similar to Policy III, the corresponding difference-differential equations can be derived under Policy IV. For convenience, we focus only on the difference between Policy III and Policy IV in terms of the difference-differential equations.

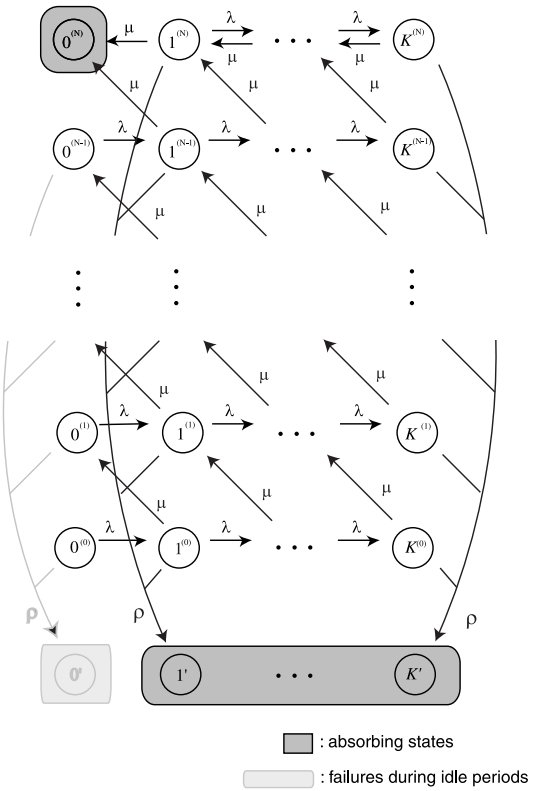

FIGURE 7. State diagram under the Policy IV.

The software rejuvenation under Policy IV is executed at the first idle period after N transactions are completed. Hence, the Markovian state corresponding to the software rejuvenation is reduced into the state $0^{(N)}$ (see Fig. 7). This difference affects the difference-differential equations in only the cases (iii) and (iv).

Case (iii): the software rejuvenation is executed,

$$\frac{\mathrm{d}p_0^{(N)}(t)}{\mathrm{d}t} = \mu(\cdot)\{p_1^{(N)}(t) + p_1^{(N-1)}(t)\}, \tag{42}$$

$$\frac{\mathrm{d}p_n^{(N)}(t)}{\mathrm{d}t} = \mu(\cdot)\{p_{n+1}^{(N)}(t) + p_{n+1}^{(N-1)}(t)\} - \{\lambda + \rho(\cdot)\}p_n^{(N)}(t), \quad n = 1, \ldots, K-1, \tag{43}$$

$$\frac{\mathrm{d}p_K^{(N)}(t)}{\mathrm{d}t} = \lambda p_{K-1}^{(N)}(t) - \mu(\cdot)p_K^{(N)}(t). \tag{44}$$

Case (iv): the server fails,

$$\frac{\mathrm{d}p_{n'}(t)}{\mathrm{d}t} = \sum_{i=0}^{N} \rho(\cdot) p_n^{(i)}(t), \quad n = 1, \ldots, K. \tag{45}$$

The difference-differential equations in the other cases are equivalent to equations (31–36). Using $p_n^{(i)}(t)$ and $p_{n'}(t)$, the probabilistic quantities needed for our analysis are given by

$$P_{AB} = \sum_{n=1}^{K} p_{n'}(\infty), \tag{46}$$

$$\mathrm{E}[U_0] = \int_0^{\infty} \sum_{i=0}^{N-1} p_0^{(i)}(t)\mathrm{d}t, \tag{47}$$

$$\mathrm{E}[U_n] = \int_0^{\infty} \sum_{i=0}^{N} p_n^{(i)}(t)\mathrm{d}t, \quad n = 1, \ldots, K, \tag{48}$$

$$\mathrm{E}[N_l] = \sum_{n=1}^{K} n p_{n'}(\infty). \tag{49}$$

Notice that, similar to equation (39), P_{AB} can be expressed as the sum of $p_{n'}(\infty)$ from $n = 1$ to K.

The difference-differential equations derived in this section can be solved with the ordinary numerical calculation methods, such as Runge–Kutta method, Adam's method and so on. It is, however, noted that the numerical calculation is not always easy to perform in the case with a large number of Markovian states, since the computation time strongly depends on the size of the underlying NHMP and usually takes much longer as the size of state is larger.

4. Numerical examples

In numerical examples, we examine the dependability measures under Policies I, II, III and IV as well as the revisited assumption.

Suppose that the failure rate is given by

$$\rho(t) = \beta\alpha t^{\alpha-1}. \tag{50}$$

The software failure occurs at the Weibull distributed random time with scale parameter β (>0) and shape parameter α (>0). Since the number of failures increases with respect to the operation time t, the shape parameter is assumed to be larger than unity, *i.e.*, the failure rate has IFR property. Let $MTTF$ denote

the mean time to failures of the software system. Then the shape parameter is given by

$$\beta = \left[\frac{\Gamma(1+1/\alpha)}{MTTF}\right]^{\alpha}, \tag{51}$$

where $\Gamma(\cdot)$ is the standard gamma function. The service rate has the monotonically increasing property and is given by

$$\mu(t) = \begin{cases} \mu_{\max} - \dfrac{t}{a}(\mu_{\max} - \mu_{\min}) & \text{for } t \le a, \\ \mu_{\min} & \text{for } t > a, \end{cases} \tag{52}$$

where a (>0) is constant. This implies that the server does not deteriorate after time a. The capacity of buffer is fixed as $K = 50$. The other model parameters are assumed as follows.

$$\begin{aligned} \lambda &= 4.0 \quad (\text{hours}^{-1}) \\ \mu_{\max} &= 12.0 \quad (\text{hours}^{-1}) \\ \mu_{\min} &= 3.0 \quad (\text{hours}^{-1}) \\ \gamma_r &= 0.85 \quad (\text{hours}) \\ \gamma_R &= 0.15 \quad (\text{hours}) \\ MTTF &= 240.0 \ (\text{hours}) \\ a &= 180.0 \ (\text{hours}). \end{aligned}$$

In the numerical examples, we call the existing models and the revisited models in this paper Model A and Model B, respectively.

Table 1 presents the optimal rejuvenation policies, T^* and N^*, maximizing the steady-state availability and minimizing the probability of transaction loss, and their associated dependability measures. From Table 1, we can find that the maximum steady-state availabilities for Model B are larger than those for Model A and that the minimum probabilities of transaction loss for Model B are always smaller than those for Model A. These results will be intuitively validated, since the probability that the server fails during the idle period is zero in Model B.

Next, we consider the case where both the failure rate and the service rate depend on the busy time of server. Let $L(t)$ denote the expected total amount of busy time, where

For Policies I and II:

$$L(t) = \int_0^t \sum_{n=1}^{K} p_n(s)\mathrm{d}s. \tag{53}$$

For Policy III:

$$L(t) = \int_0^t \sum_{n=1}^{K} \sum_{i=0}^{N-1} p_n^{(i)}(s)\mathrm{d}s. \tag{54}$$

TABLE 1. Operation time dependent case.

Policy &		maximization of availability				minimization of loss probability			
Model	α	Optimal	$A_{\rm ss}$	$P_{\rm loss}$	$T_{\rm res}$	Optimal	$A_{\rm ss}$	$P_{\rm loss}$	$T_{\rm res}$
I-A	1.5	160.58	0.9969	1.308e-02	0.3677	92.38	0.9967	6.473e-03	0.1830
	2.0	127.66	0.9976	6.770e-03	0.2391	90.84	0.9974	5.716e-03	0.1825
	2.5	125.22	0.9980	6.239e-03	0.2358	91.42	0.9978	5.319e-03	0.1838
I-B	1.5	136.72	0.9979	5.817e-03	0.1764	95.34	0.9977	4.613e-03	0.1251
	2.0	132.48	0.9982	5.191e-03	0.1696	95.26	0.9980	4.236e-03	0.1254
	2.5	133.40	0.9984	5.031e-03	0.1723	96.16	0.9982	4.023e-03	0.1264
II-A	1.5	131.00	0.9969	5.987e-03	0.3450	100.96	0.9968	3.682e-03	0.1950
	2.0	121.52	0.9976	3.082e-03	0.2405	98.16	0.9975	2.806e-03	0.1930
	2.5	120.40	0.9980	2.488e-03	0.2376	101.78	0.9979	2.301e-03	0.1988
II-B	1.5	126.10	0.9978	3.615e-03	0.2782	103.16	0.9978	2.760e-03	0.2004
	2.0	124.16	0.9981	2.750e-03	0.2591	104.06	0.9981	2.274e-03	0.2026
	2.5	124.50	0.9983	2.425e-03	0.2610	107.32	0.9983	1.973e-03	0.2084
III-A	1.5	634	0.9969	1.351e-02	0.3802	368	0.9967	6.476e-03	0.1833
	2.0	509	0.9976	6.816e-03	0.2411	362	0.9974	5.720e-03	0.1829
	2.5	499	0.9980	6.274e-03	0.2375	364	0.9978	5.323e-03	0.1841
III-B	1.5	539	0.9979	7.594e-03	0.2641	365	0.9976	5.758e-03	0.1841
	2.0	526	0.9982	6.892e-03	0.2563	365	0.9979	5.344e-03	0.1846
	2.5	530	0.9984	6.765e-03	0.2607	368	0.9981	5.115e-03	0.1858
IV-A	1.5	633	0.9969	4.381e-03	0.3761	412	0.9968	3.677e-03	0.1959
	2.0	508	0.9976	2.941e-03	0.2401	397	0.9975	2.798e-03	0.1928
	2.5	498	0.9980	2.394e-03	0.2365	412	0.9979	2.297e-03	0.1986
IV-B	1.5	543	0.9979	2.960e-03	0.2668	420	0.9978	2.751e-03	0.2010
	2.0	526	0.9982	2.424e-03	0.2557	422	0.9981	2.267e-03	0.2026
	2.5	529	0.9983	2.100e-03	0.2593	437	0.9983	1.967e-03	0.2089

For Policy IV:

$$L(t) = \int_0^t \sum_{n=1}^{K} \sum_{i=0}^{N} p_n^{(i)}(s) \mathrm{d}s. \tag{55}$$

In the busy time dependent case, the failure rate and the service rate are represented as $\rho(L(t))$ and $\mu(L(t))$, respectively. The other model parameters are same as those in the operation dependent case.

Table 2 presents the optimal rejuvenation policies and their associated dependability measures in this case. These results give us the similar tendency of the dependability measures in the operation time dependent case. That is, Model A tends to underestimate the steady-state availability and to overestimate the probability of transaction loss, comparing with Model B. These tendencies are observed in the operation dependent case. Note that, however, Model A tends to underestimate the upper bound of the mean response time under the maximization of the steady-state availability. This is a remarkably different result from the operation time dependent case.

Finally, we investigate the case where the failure and service rates depend on the system workload. The failure and service rates are given by $\rho(W(t))$ and $\mu(W(t))$, where $W(t)$ is the expected total amount of workload before time t.

TABLE 2. Busy time dependent case.

Policy &		maximization of availability				minimization of loss probability			
Model	α	Optimal	$A_{\rm ss}$	$P_{\rm loss}$	$T_{\rm res}$	Optimal	$A_{\rm ss}$	$P_{\rm loss}$	$T_{\rm res}$
I-A	1.5	234.92	0.9979	3.240e-03	0.1674	256.24	0.9979	3.233e-03	0.1726
	2.0	206.44	0.9985	2.640e-03	0.1664	209.64	0.9985	2.640e-03	0.1673
	2.5	216.42	0.9989	2.350e-03	0.1727	203.94	0.9989	2.341e-03	0.1682
I-B	1.5	250.48	0.9987	2.782e-03	0.1830	212.46	0.9987	2.716e-03	0.1687
	2.0	241.70	0.9990	2.467e-03	0.1842	203.78	0.9989	2.378e-03	0.1684
	2.5	247.62	0.9991	2.346e-03	0.1901	204.16	0.9991	2.203e-03	0.1696
II-A	1.5	234.02	0.9979	2.476e-03	0.1675	181.52	0.9978	2.439e-03	0.1557
	2.0	205.72	0.9985	1.639e-03	0.1665	177.02	0.9985	1.619e-03	0.1583
	2.5	215.60	0.9989	1.245e-03	0.1729	192.24	0.9989	1.230e-03	0.1646
II-B	1.5	249.06	0.9987	1.733e-03	0.1831	196.12	0.9987	1.677e-03	0.1639
	2.0	240.18	0.9990	1.290e-03	0.1843	199.58	0.9989	1.247e-03	0.1673
	2.5	245.90	0.9991	1.054e-03	0.1902	212.22	0.9991	1.020e-03	0.1734
III-A	1.5	940	0.9979	2.128e-03	0.1673	1028	0.9979	3.221e-03	0.1728
	2.0	825	0.9985	1.458e-03	0.1662	842	0.9985	2.622e-03	0.1677
	2.5	864	0.9989	1.126e-03	0.1725	822	0.9989	2.319e-03	0.1688
III-B	1.5	1001	0.9987	2.754e-03	0.1831	856	0.9987	2.696e-03	0.1693
	2.0	966	0.9990	2.433e-03	0.1843	821	0.9989	2.355e-03	0.1690
	2.5	990	0.9991	2.304e-03	0.1903	824	0.9991	2.177e-03	0.1705
IV-A	1.5	938	0.9979	2.475e-03	0.1673	731	0.9978	2.438e-03	0.1557
	2.0	825	0.9985	1.639e-03	0.1663	710	0.9985	1.619e-03	0.1581
	2.5	865	0.9989	1.245e-03	0.1727	770	0.9989	1.229e-03	0.1643
IV-B	1.5	1002	0.9987	1.733e-03	0.1831	790	0.9987	1.675e-03	0.1639
	2.0	966	0.9990	1.290e-03	0.1842	802	0.9989	1.245e-03	0.1672
	2.5	991	0.9991	1.055e-03	0.1903	852	0.9991	1.018e-03	0.1732

For Policies I and II:

$$W(t) = \int_0^t \sum_{n=1}^{K} \mu(s) p_n(s) \mathrm{d}s. \tag{56}$$

For Policy III:

$$W(t) = \int_0^t \sum_{n=1}^{K} \sum_{i=0}^{N-1} \mu(s) p_n^{(i)}(s) \mathrm{d}s. \tag{57}$$

For Policy IV:

$$W(t) = \int_0^t \sum_{n=1}^{K} \sum_{i=0}^{N} \mu(s) p_n^{(i)}(s) \mathrm{d}s. \tag{58}$$

The optimal rejuvenation policies and their associated dependability measures in the workload dependent case are presented in Table 3. In the workload dependent case, Model A also tends to underestimate the steady-state availability and to overestimate the probability of transaction loss. In the maximization of the

TABLE 3. Workload dependent case.

Policy &		maximization of availability				minimization of loss probability			
Model	α	Optimal	A_{ss}	P_{loss}	T_{res}	Optimal	A_{ss}	P_{loss}	T_{res}
I-A	1.5	137.86	0.9954	1.658e-01	6.3716	26.04	0.9925	1.956e-02	0.1935
	2.0	75.60	0.9954	1.178e-01	2.4496	25.50	0.9931	1.903e-02	0.1923
	2.5	61.50	0.9955	9.353e-02	1.3456	25.34	0.9935	1.871e-02	0.1922
I-B	1.5	99.70	0.9958	1.471e-01	4.5027	25.48	0.9934	1.906e-02	0.1924
	2.0	65.24	0.9958	1.031e-01	1.6706	25.30	0.9936	1.873e-02	0.1921
	2.5	56.78	0.9958	8.392e-02	1.0502	25.28	0.9938	1.854e-02	0.1923
II-A	1.5	43.08	0.9954	1.621e-01	7.8012	34.34	0.9938	7.140e-03	0.3010
	2.0	39.28	0.9953	1.061e-01	4.9254	33.52	0.9944	6.368e-03	0.2949
	2.5	37.94	0.9955	7.637e-02	3.6149	33.32	0.9948	5.841e-03	0.2978
II-B	1.5	40.46	0.9958	1.445e-01	6.9098	33.00	0.9946	6.292e-03	0.2857
	2.0	38.20	0.9957	9.259e-02	4.3411	33.14	0.9949	5.820e-03	0.2952
	2.5	37.36	0.9957	6.754e-02	3.2687	33.14	0.9951	5.475e-03	0.2991
III-A	1.5	454	0.9954	1.658e-01	6.3861	106	0.9927	1.888e-02	0.1978
	2.0	266	0.9954	1.179e-01	2.5456	103	0.9932	1.832e-02	0.1956
	2.5	223	0.9955	9.334e-02	1.4412	103	0.9936	1.800e-02	0.1963
III-B	1.5	338	0.9958	1.472e-01	4.5479	103	0.9935	1.835e-02	0.1958
	2.0	234	0.9958	1.031e-01	1.7721	103	0.9938	1.802e-02	0.1965
	2.5	209	0.9958	8.410e-02	1.1540	102	0.9939	1.781e-02	0.1955
IV-A	1.5	452	0.9954	1.106e-01	5.9816	156	0.9941	6.903e-03	0.3103
	2.0	263	0.9954	2.524e-02	2.2554	150	0.9946	6.173e-03	0.2969
	2.5	222	0.9955	1.238e-02	1.3179	148	0.9950	5.658e-03	0.2945
IV-B	1.5	337	0.9958	6.324e-02	0.4145	150	0.9950	6.033e-03	0.3000
	2.0	233	0.9958	1.506e-02	1.6098	148	0.9952	5.589e-03	0.2963
	2.5	208	0.9958	9.598e-03	1.0612	148	0.9954	5.251e-03	0.2987

steady-state availability, we can find the noticeable difference between Model A and Model B, that is, the optimal policies are quite different from each other. By contrast, in the minimization of the probability of transaction loss, the optimal policies in both cases are quite similar.

5. Conclusion

In this paper, we have revisited the software rejuvenation models. By making the plausible assumption that the server never fails during idle periods, we have reformulated three dependability measures under four kinds of rejuvenation policies. Furthermore, we have compared numerically the dependability measures under both the existing and the revisited assumptions. As a result, it has been found that Garg *et al.* [6] model tends to underestimate and overestimate the steady-state availability and the probability of transaction loss, respectively, and

makes pessimistic decisions on the determination of the optimal software rejuvenation policy.

Acknowledgements. This work is supported in part by Grant-in-Aid for Scientific Research from the Ministry of Education, Science and Culture of Japan under Grant No. 13480109 (2001–2003).

References

[1] E. Adams, Optimizing preventive service of the software products. *IBM J. Res. Development* **28** (1984) 2-14.
[2] A. Avritzer and E.J. Weyuker, Monitoring smoothly degrading systems for increased dependability. *Empirical Software Engrg.* **2** (1997) 59-77.
[3] T. Dohi, K. Goševa–Popstojanova and K.S. Trivedi, Estimating software rejuvenation schedule in high assurance systems. *Comput. J.* **44** (2001) 473-485.
[4] S. Garg, A. Puliafito, M. Telek and K.S. Trivedi, Analysis of software rejuvenation using Markov regenerative stochastic Petri net, in *Proc. 6th Int'l Symp. on Software Reliability Eng.* IEEE CS Press, Los Alamitos (1995) 24-27.
[5] S. Garg, Y. Huang, C. Kintala and K.S. Trivedi, Time and load based software rejuvenation: Policy, evaluation and optimality, in *Proc. 1st Fault-Tolerant Symp.* (1995) 22-25.
[6] S. Garg, S. Pfening, A. Puliafito, M. Telek and K.S. Trivedi, Analysis of preventive maintenance in transactions based software systems. *IEEE Trans. Comput.* **47** (1998) 96-107.
[7] J. Gray and D.P. Siewiorek, High-availability computer sysmtes. *IEEE Comput.* **24** (1991) 39-48.
[8] D.P. Heyman, Optimal operating policies for *M/G/1* queueing system. *Oper. Res.* **16** (1968) 362-382.
[9] D.P. Heyman, The *T*-policy for the *M/G/1* queue. *Management Sci.* **23** (1977) 775-778.
[10] Y. Huang, C. Kintala, N. Kolettis and N.D. Fulton, Software rejuvenation: Analysis, module and applications, in *Proc. 25th Int'l Symp. on Fault Tolerant Computing.* IEEE CS Press, Los Alamitos (1995) 381-390.
[11] V.G. Kulkarni, *Modeling, Analysis, Design, and Control of Stochastic Systems*. Springer-Varlag, New York (1999).
[12] H. Okamura, S. Miyahara, T. Dohi and S. Osaki, Performance evaluation of workload-based software rejuvenation scheme. *IEICE Trans. Inform. Systems* **E84D** (2001) 1368-1375.
[13] D.L. Parnas, Software aging, in *Proc. 16th Int'l Conf. on Software Eng.* ACM, New York (1994) 279-287.
[14] A.T. Tai, L. Alkalai and S.N. Chau, On-board preventive maintenance for long-life deep space missions: A model – based analysis, in *Proc. 3rd IEEE Int'l Computer Performance and Dependability Symp.* IEEE CS Press, Los Alamitos (1998) 196-205.
[15] A.T. Tai, L. Alkalai and S.N. Chau, On-board preventive maintenance: A design-oriented analytic study for long-life applications. *Performance Evaluation* **35** (1999) 215-232.