数理解析研究所講究録 1409 巻 2005 年 1-9

Discrete Optimal Testing/Maintenance Policy in a Software Development Project

林坂弘一郎, 土肥 正
Koichiro Rinsaka and Tadashi Dohi
Department of Information Engineering, Graduate School of Engineering,
Hiroshima University, Japan

1 Introduction

It is important to determine the optimal time when software testing should be stopped and when the system should be delivered to a user or a market. This problem, called *optimal software release problem*, plays a central role for the success or failure of a software development project. Okumoto and Goel [1] assumed that the number of software faults detected in the testing phase is described by an exponential software reliability model based on a non-homogeneous Poisson process (NHPP) [2], and derived an optimal software release time which minimizes the total expected software cost. Koch and Kubat [3] considered the similar problem for the other software reliability model by Jelinski and Moranda [4]. Bai and Yun [5] calculated the optimal number of faults detected before the release under the Jelinski and Moranda model. Many authors formulated the optimal software release problems based on different model assumptions and/or several software reliability models [6, 7].

It is difficult to detect and remove all faults remaining in a software during the actual testing phase, because exhaustive testing of all executable paths in a general program is impossible. Once the software is released to users, however the software failures may occur even in the operational phase. It is common for software developers to provide maintenance service during the period when they are still responsible for fixing software faults causing failures. In order to carry out the maintenance in the operational phase, the software developer has to keep a software maintenance team. At the same time, the management cost in the operational phase should be reduced as much as possible, but at the same time the human resources should be utilized effectively. Although the problem which determines the maintenance period is important from the practical point of view, only a very few authors paid their attention to this problem.

Kimura *et al.* [8] considered the optimal software release problem in the case where the software warranty period is a random variable. Pham and Zhang [9] developed a software cost model with both warranty and risk. They focused on the problem for determining when to stop the software testing under a warranty contract. However, it is noted that the software developer has to design the warranty contract itself and often provides the posterior service for users after software failures. Dohi *et al.* [10] formulated the problem for determining the optimal software warranty period which minimizes the total expected software cost under the assumption that the debugging process in the testing phase is described by an NHPP. Sandoh and Rinsaka [11] considered the design problem of a maintenance service contract by regarding as the Stackelberg game between a software agent and a software user. Since the user's operational environment is not always same as that assumed in the software development phase, however, the above literature did not take account of the difference between two different phases.

Several reliability assessment methods during the operational phase have been proposed by some authors [12, 13]. Rinsaka and Dohi [14] developed a continuous time model for designing the optimal testing and maintenance periods, where the difference between the software testing environment and the operational environment are reflected.

In this paper, we focus on the optimal software testing/maintenance problem considered in Rinsaka and Dohi [14], and develop a stochastic model in discrete circumstance, where the difference between the software testing environment and the operational environment can be characterized by an environment factor (see Okamura *et al.* [13]). More precisely, the total expected software cost is formulated via the discrete NHPP type of software reliability models [16, 17, 18]. In the special case with the geometric fault-detection time distribution, we derive analytically the optimal testing period (release time) which minimizes the total expected software cost under a milder condition. We call the time length to complete the operational maintenance after the release *a planned maintenance limit*, and also derive the optimal planned maintenance limit which minimizes the total expected software cost. In numerical examples with real data, we calculate numerically the joint optimal testing/maintenance policy, combined by testing period and planned maintenance limit.

2 Model Description

First, we make the following assumptions on the software fault-detection process:

(a) In each time when a software failure occurs, the software fault causing the failure can be detected and removed immediately.

(b) The number of initial faults contained in the software program, N_0, is given by the Poisson distributed random variable with mean ω (> 0).

(c) The time to detect each software fault is independent and identically distributed nonnegative discrete random variable with probability mass function p_i ($i = 0, 1, 2, \cdots$) and probability distribution function $P(i) = \sum_{k=1}^{i} p_k$, where $0 \leq p_i \leq 1$ and $0 \leq P(i) \leq 1$.

Let $\{N_i,\ i = 1, 2, \cdots\}$ be the cumulative number of software faults detected up to time i. From the above assumptions, the probability mass function of N_i is given by

$$\begin{aligned} \Pr\{N_i = m\} &= \frac{\{\omega \sum_{k=1}^{i} p_k\}^m}{m!} e^{-\omega \sum_{k=1}^{i} p_k} \\ &= \frac{\{\omega P(i)\}^m}{m!} e^{-\omega P(i)} \qquad m = 0, 1, 2, \cdots. \end{aligned} \tag{1}$$

Hence, the stochastic process $\{N_i,\ i = 1, 2, \cdots\}$ is equivalent to a discrete time NHPP with mean value function $\omega P(i)$.

Suppose that a software testing is started at time $i = 0$ and terminated at time $i = n_0$ (≥ 0). After completing the software testing, the software product is released to a user or the market. The time length of software life cycle n_L (> 0) is a known constant in advance and is assumed to be sufficiently larger than n_0. More precisely, the software life cycle is measured from the point of time n_0. The software developer is responsible to the maintenance service for all the software failures that may occur during the software life cycle under a maintenance contract. We suppose that the project manager decides to break up the maintenance team at time $n_0 + n_W$ for reduction of the operational cost to keep it, but a large amount of debugging cost during $(n_0 + n_W, n_0 + n_L]$ may be needed if the software failure occurs. Let n_W be the planned maintenance limit denoting the time length to complete the operational maintenance after the release *a planned maintenance limit*. Further, we define the following cost components:

c_0 (> 0): cost to remove each fault in the testing phase,

c_W (> 0): cost to remove each fault before the planned maintenance limit,

c_L (> 0): cost to remove each fault after the planned maintenance limit,

k_0 (> 0): testing cost per unit of time,

k_W (> 0): operational cost to keep the maintenance team per unit of time.

In the following section, we formulate the total expected software cost by introducing the above cost factors in testing and operational phases. We derive the discrete optimal software testing period n_0^* or the discrete optimal planned maintenance limit n_W^* which minimizes the total expected software cost at the end of the software life cycle. Then, we calculate the joint optimal policy (n_0^{**}, n_W^{**}) combined by both testing period and planned maintenance limit.

3 Total Expected Software Cost

We formulate the total expected software cost which can occur in both testing and operational phases. In the operational phase, we consider two cost factors; the maintenance cost due to the software failure and the operational cost to keep the maintenance team.

From Eq.(1), the probability mass function of the number of software faults detected during the testing phase is given by

$$\Pr\{N_{n_0} = m\} = \frac{\{\omega P(n_0)\}^m}{m!} e^{-\omega P(n_0)}. \tag{2}$$

It should be noted that the operational environment after the release may differ from the debugging environment in the testing phase. This difference is similar to that between the accelerated life testing environment and the normal operating environment for hardware products. We introduce the environment factor a (> 0) which represents the relative severity in the operational environment after the release, and assume that the times in the testing phase and the operational phase have a proportional relationship. Namely, the time length n in the operational phase corresponds to $a \times n$ in the testing phase. Under the above assumption, $a = 1$ means the equivalence between the testing and operational environments. On the other hand, $a > 1$ ($a < 1$) implies that the operational environment is severer (looser) than the testing environment. Okamura *et al.* [13] apply this technique to model the operational phase of the software, and estimate the software reliability through an example in the actual software development project. The probability mass function of the number of software faults detected before the planned maintenance limit is given by

$$\Pr\{N_{n_0+n_W} - N_{n_0} = m\} = \frac{\{\omega\{P(n_0+[an_W]) - P(n_0)\}\}^m}{m!} e^{-\omega\{P(n_0+[an_W])-P(n_0)\}}, \tag{3}$$

where, $[\cdot]$ is the Gaussian integer.

Similarly, the fault-detection process of the software after the planned maintenance limit is expressed by

$$\begin{aligned}&\Pr\{N_{n_0+n_L} - N_{n_0+n_W} = m\}\\ &= \frac{\{\omega\{P(n_0+[an_L]) - P(n_0+[an_W])\}\}^m}{m!} e^{-\omega\{P(n_0+[an_L])-P(n_0+[an_W])\}}.\end{aligned} \tag{4}$$

From Eqs.(2), (3) and (4), the total expected software cost is given by

$$\begin{aligned}C(n_0, n_W) &= k_0 n_0 + c_0\omega P(n_0) + k_W n_W + c_W\omega\{P(n_0+[an_W]) - P(n_0)\}\\ &\quad + c_L\omega\{P(n_0+[an_L]) - P(n_0+[an_W])\}.\end{aligned} \tag{5}$$

4 Determination of the Optimal Policies

In this section we derive the optimal testing period n_0^* or the optimal planned maintenance limit n_W^* which minimizes the total expected software cost incurred to the software developer at the end of software life cycle. Suppose that the time to detect each software fault obeys the geometric distribution

$$p_i = b(1-b)^{i-1} \tag{6}$$

with parameter b ($0 < b < 1$). In this case, the total expected software cost in Eq.(5) becomes

$$\begin{aligned}C(n_0, n_W) &= k_0 n_0 + c_0\omega\{1-(1-b)^{n_0}\} + k_W n_W + c_W\omega\left\{(1-b)^{n_0} - (1-b)^{n_0+[an_W]}\right\}\\ &\quad + c_L\omega\left\{(1-b)^{n_0+[an_W]} - (1-b)^{n_0+[an_L]}\right\}.\end{aligned} \tag{7}$$

We make the following assumptions:

(A-I) a is positive and an integer value,

(A-II) $c_L > c_W > c_0$,

(A-III) $c_W\{1-(1-b)^{an_L}\} > c_0$,

(A-IV) $c_W\{1-(1-b)^{an_W}\} + c_L\{(1-b)^{an_W} - (1-b)^{an_L}\} > c_0$.

Define

$$Q(n_W) = k_0 + \omega b\{c_0 - c_W(1-(1-b)^{an_W}) - c_L((1-b)^{an_W} - (1-b)^{an_L})\}. \tag{8}$$

Then the following result provides the optimal software testing policy which minimizes the total expected software cost.

Theorem 1: *When the software fault-detection time distribution follows the geometric distribution with parameter b* $(0 < b < 1)$*, under the assumptions (A-I) to (A-IV), the optimal software testing period (release time) which minimizes the total expected software cost is given as follows:*

(1) If $Q(n_W) < 0$*, then there exist (at least one, at most two) finite optimal software testing periods (release times)* n_0^* (> 0)*.*

(2) If $Q(n_W) \geq 0$*, then the optimal policy is* $n_0^* = 0$ *with*

$$C(n_0^*, n_W) = k_W n_W + c_W \omega \left\{1 - (1-b)^{an_W}\right\} + c_L \omega \left\{(1-b)^{an_W} - (1-b)^{an_L}\right\}. \tag{9}$$

Furthermore, the following result provides the optimal planned maintenance limit which minimizes the total expected software cost.

Theorem 2: *When the software fault-detection time distribution follows the geometric distribution with parameter b* $(0 < b < 1)$*, under the assumptions (A-I) and (A-II), the optimal planned maintenance limit which minimizes the total expected software cost is given as follows:*

(1) If $k_W \geq (c_L - c_W)\omega \left\{1 - (1-b)^a\right\}(1-b)^{n_0}$*, then the optimal policy is* $n_W^* = 0$ *with*

$$C(n_0, n_W^*) = k_0 n_0 + c_0 \omega \left\{1 - (1-b)^{n_0}\right\} + c_W \omega \left\{(1-b)^{n_0} - (1-b)^{n_0 + an_L}\right\}. \tag{10}$$

(2) If $k_W < (c_L - c_W)\omega \left\{1 - (1-b)^a\right\}(1-b)^{n_0}$ *and* $k_W > (c_L - c_W)\omega \left\{1 - (1-b)^a\right\}(1-b)^{n_0 + a(n_L - 1)}$*, then there exist (at least one, at most two) optimal planned maintenance limits* n_W^* $(0 < n_W^* < n_L)$ *which minimizes the total expected software cost.*

(3) If $k_W \leq (c_L - c_W)\omega \left\{1 - (1-b)^a\right\}(1-b)^{n_0 + a(n_L - 1)}$*, then we have* $n_W^* = n_L$ *with*

$$C(n_0, n_W^*) = k_0 n_0 + c_0 \omega \left\{1 - (1-b)^{n_0}\right\} + k_W n_L + c_W \omega \left\{(1-b)^{n_0} - (1-b)^{n_0 + an_L}\right\}. \tag{11}$$

5 Numerical Examples

Based on 351 software fault (41 week) data observed in the real software testing process [19], we calculate numerically the optimal testing period n_0^* and the optimal planned maintenance limit n_W^* which minimize the total expected software cost. Further, we compute the joint optimal policy (n_0^{**}, n_W^{**}) minimizing $C(n_0, n_W)$. For the software fault-detection time distribution, we apply three probability distributions; geometric, negative binomial and discrete Weibull distributions. The probability mass functions for negative binomial and discrete Weibull are given by

$$p_i = \binom{h+i-2}{h-1} b^h (1-b)^{i-1}, \tag{12}$$

and

$$p_i = b^{i^h} - b^{(i+1)^h}, \tag{13}$$

respectively, where $h \geq 0$. For negative binomial and discrete Weibull distributions, we consider the case of $h = 2$.

Suppose that the unknown parameters in the software reliability models are estimated by the method of maximum likelihood. Then, we have the estimates $(\hat{\omega}, \hat{b}) = (413.305,\ 0.0451012)$ for the geometric model, $(\hat{\omega}, \hat{b}) = (364.234,\ 0.116255)$ for the negative binomial model and $(\hat{\omega}, \hat{b}) = (351.871,\ 0.996436)$ for the discrete Weibull model. Figure 1 shows the actual software fault data and the behavior of estimated mean value functions. Since the environment factor a is a subjective parameter which should be estimated from the past development track record, we assume that the value of a is known. For the other model parameters, we assume: $k_0 = 2.0$, $k_W = 1.0$, $c_0 = 5.0$, $c_W = 10.0$, $c_L = 50.0$ and $n_L = 200$.

Table 1 presents the dependence of environment factor a on the optimal testing period n_0^* when $n_W = 20$. As the environment factor monotonically increases, *i.e.*, the operational circumstance tends to be severe, it is observed that the optimal testing period n_0^* and its associated minimum total expected software cost $C(n_0^*, 20)$ decrease for both geometric and negative binomial models. For the discrete Weibull model, since it is estimated that the software fault is hardly discovered after the release as shown

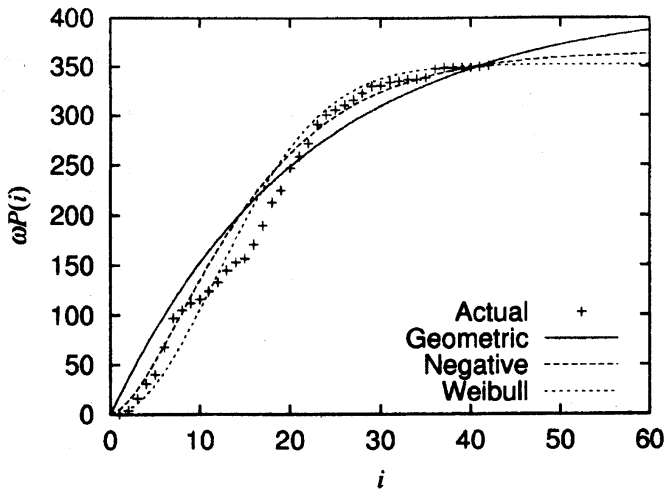

Figure 1: Behavior of actual software fault data and estimated mean value functions.

Table 1: Optimal software testing period for varying environment factor.

	Geometric		Negative		Weibull	
a	n_0^*	$C(n_0^*, 20)$	n_0^*	$C(n_0^*, 20)$	n_0^*	$C(n_0^*, 20)$
0.50	122	2375	65	1990	41	1867
0.75	119	2367	62	1983	40	1865
1.00	115	2359	59	1977	39	1865
1.25	111	2352	57	1973	39	1865
1.50	108	2345	56	1971	39	1865
2.00	101	2333	54	1967	39	1865
3.00	93	2315	53	1966	39	1865

Table 2: Optimal planned maintenance limit for varying environment factor.

	Geometric		Negative		Weibull	
a	n_W^*	$C(41, n_W^*)$	n_W^*	$C(41, n_W^*)$	n_W^*	$C(41, n_W^*)$
0.50	176	2648	62	2050	12	1863
0.75	128	2615	48	2028	8	1859
1.00	103	2584	38	2016	8	1856
1.25	88	2564	32	2008	8	1855
1.50	74	2549	28	2003	6	1854
2.00	59	2530	22	1996	5	1852
3.00	42	2509	16	1988	4	1850

Table 3: Optimal testing period/planned maintenance limit for varying environment factor.

	Geometric			Negative			Weibull		
a	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$
0.50	131	0	2372	73	0	1986	47	0	1859
0.75	108	40	2365	64	16	1983	42	8	1858
1.00	99	45	2352	60	19	1977	42	7	1856
1.25	94	44	2343	57	20	1973	41	8	1855
1.50	93	40	2335	57	16	1970	41	6	1854
2.00	90	34	2324	56	14	1966	41	5	1852
3.00	88	27	2312	55	11	1960	40	4	1850

in Fig.1, it is observed that the optimal testing period is strongly influenced by varying environment factor.

Table 2 shows the dependence of environment factor a on the optimal planned maintenance limit n_W^* in case of $n_0 = 41$. It is found that the optimal planned maintenance limit n_W^* and the corresponding minimum total expected software cost $C(41, n_W^*)$ decrease as the environment factor monotonically increases. This tendency can be explained as follows: The residual faults in software are detected and removed at the early stage in the operational phase as the operational environment becomes more severe. Then, the possibility that the software failure occurs in the latter stage of the operational phase may become small. Hence, the implication in which the software developer keeps the maintenance team becomes smaller toward the end of the life cycle.

Table 3 examines the dependence of environment factor a on the joint optimal policy (n_0^{**}, n_W^{**}) combined by testing period and planned maintenance limit. Figure 2 illustrates the behavior of the expected cost for the geometric model when $a = 2.00$. It is observed from Table 3 that the optimal testing period n_0^{**} decreases as the environment factor monotonically increases, but, the monotonicity of the optimal planned maintenance limit n_W^{**} is not observed. It is also seen that the minimum total expected software cost $C(n_0^{**}, n_W^{**})$ decreases as the environment factor monotonically increases.

Tables 4, 5 and 6 present the dependence of the software reliability model parameter b on the joint optimal policy (n_0^{**}, n_W^{**}). It is observed that the optimal testing time, optimal planned maintenance limit and its associated minimum total expected software cost decrease as the fault detection becomes easier.

6 Concluding Remarks

In this paper, we have assumed that the software developer was responsible to the maintenance service for all the software failures that occur during the software life cycle under the maintenance contract. In order to carry out the maintenance service in the operational phase, the software developer has to keep a software maintenance team. At the same time, the management cost in the operational phase has to be reduced as much as possible, but human resources should be utilized effectively. We have called the time length to complete the operational maintenance after the release the planned maintenance limit, and have controlled it in terms of cost-benefit analysis. We have developed the discrete model which represents the difference in the software execution environment during testing and operational phases, using the same method as the continuous-time-based reliability assessment modeling in the operational phase proposed by Okamura *et al.* [13]. Based on the discrete NHPP, we have formulated the total expected software cost incurred to the software developer at the end of software life cycle. The optimal testing period (release time) and optimal planned maintenance limit which minimize the total expected software cost have been derived. Then, throughout the numerical examples, we have discussed the joint optimal policy combined by testing period and planned maintenance limit.

Table 4: Optimal testing period/planned maintenance limit for geometric software reliability model with varying parameter b.

	Geometric		
b	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$
0.042	95	37	2340
0.043	93	36	2335
0.044	92	35	2330
0.045	90	35	2325
0.046	88	34	2320
0.047	87	33	2315
0.048	86	32	2311

Table 5: Optimal testing period/planned maintenance limit for negative binomial software reliability model with varying parameter b.

	Negative		
b	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$
0.08	77	22	2025
0.09	70	19	2004
0.10	64	17	1987
0.11	59	15	1973
0.12	54	14	1961
0.13	50	13	1951
0.14	47	12	1942

Table 6: Optimal testing period/planned maintenance limit for discrete Weibull software reliability model with varying parameter b.

	Weibull		
b	n_0^{**}	n_W^{**}	$C(n_0^{**}, n_W^{**})$
0.999	72	10	1926
0.998	53	7	1880
0.997	44	6	1860
0.996	38	5	1847
0.995	35	4	1838
0.994	32	4	1832
0.993	30	4	1827

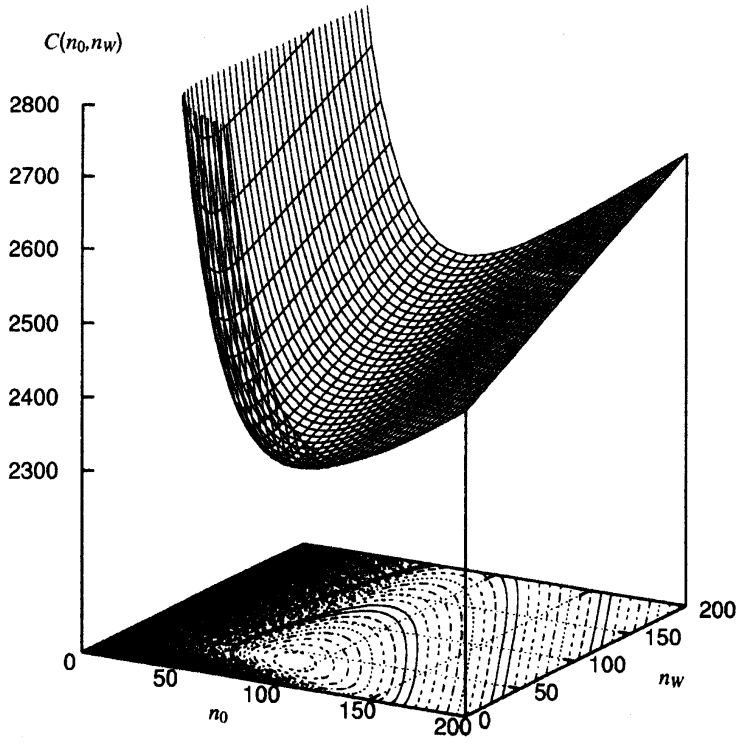

Figure 2: Behavior of the total expected software cost for geometric software reliability model ($a = 2.00$).

References

[1] K. Okumoto and L. Goel, "Optimum release time for software systems based on reliability and cost criteria," J. Sys. Software, vol. 1, pp. 315–318, 1980.

[2] A.L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Trans. Reliab., vol. R-28, no. 3, pp. 206–211, 1979.

[3] H.S. Koch and P. Kubat, "Optimal release time of computer software," IEEE Trans. Software Eng., vol. SE-9, no. 3, pp. 323–327, 1983.

[4] Z. Jelinski and P.B. Moranda, "Software reliability research," Statistical Computer Performance Evaluation, (W. Freiberger ed.), pp. 465–484, Academic Press, New York, 1972.

[5] D.S. Bai and W.Y. Yun, "Optimum number of errors corrected before releasing a software system," IEEE Trans. Reliab., vol. R-37, no. 1, pp. 41–44, 1988.

[6] W.Y. Yun and D.S. Bai, "Optimum software release policy with random life cycle," IEEE Trans. Reliab., vol. R-39, no. 2, pp. 167–170, 1990.

[7] T. Dohi, N. Kaio and S. Osaki, "Optimal software release policies with debugging time lag," Int. J. Reliab., Quality and Safety Eng., vol. 4, no. 3, pp. 241–255, 1997.

[8] M. Kimura, T. Toyota and S. Yamada, "Economic analysis of software release problems with warranty cost and reliability requirement," Reliab. Eng. & Sys. Safe., vol. 66, no. 1, pp. 49–55, 1999.

[9] H. Pham and X. Zhang, "A software cost model with warranty and risk costs," IEEE Trans. Comput., vol. 48, no. 1, pp. 71–75, 1999.

[10] T. Dohi, H. Okamura, N. Kaio and S. Osaki, "The age-dependent optimal warranty policy and its application to software maintenance contract," Proc. 5th Int'l Conf. on Probab. Safe. Assess. and Mgmt. (S. Kondo and K. Furuta, eds.), vol. 4, pp. 2547–2552, University Academy Press Inc., 2000.

[11] H. Sandoh and K. Rinsaka, "Maintenance service contract model for software," Proc. of the First Western Pacific and Third Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management, Christchurch, New Zealand, pp. 466–475, 1999.

[12] J. Musa, G. Fuoco, N. Irving, D. Kropfl and B. Juhlin, "The operational profile," Handbook of Software Reliability Engineering, (M.R. Lyu ed.), pp. 167–216, McGraw-Hill, New York, 1995.

[13] H. Okamura, T. Dohi and S. Osaki, "A reliability assessment method for software products in operational phase — proposal of an accelerated life testing model —," Electronics and Communication in Japan, Part 3, vol. 84, pp. 25–33, 2001.

[14] K. Rinsaka and T. Dohi, "Optimal testing/maintenance design in a software development project," Electronic Proc. of Fourth International Conference on Mathematical Methods in Reliability – Methodology and Practice, Santa Fe, New Mexico, USA, 2004.

[15] A.A. Abde-Ghaly, P.Y. Chan and B. Littlewood, "Evaluation of competing software reliability predictions," IEEE Trans. Software Eng., vol. SE-12, no. 9, pp. 950–967, 1986.

[16] S. Yamada and S. Osaki, "Discrete software reliability growth models," Applied Stochastic Models and Data Analysis, vol. 1, pp. 65–77, 1985.

[17] T. Kitaoka, S. Yamada and S. Osaki, "A discrete non-homogeneous error detection rate model for software reliability," Transactions of the Institute of Electronics and Communication Engineers of Japan, vol. E69, pp. 859–865, 1986.

[18] H. Okamura, A. Murayama and T. Dohi, "EM Algorithm for dis-crete software reliability models: a unified parameter estimation method," Proceedings of 8th IEEE International Symposium on High Assurance Systems Engineering, pp. 219–228. IEEE CS Press, 2004.

[19] A.P. Nikora and M. R. Lyu, "Software reliability measurement experience," Handbook of Software Reliability Engineering, (M. R. Lyu ed.), pp. 255–301, McGraw-Hill, New York, 1995.