Proceedings of International Conference on Distributed
Systems, Software Engineering and Database Systems

Cost–Reliability Optimal Software Release Policy under Penalty Cost Based on the HGDM

Rong-Huei Hou Sy-Yen Kuo Yi-Ping Chang[1]

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.
Email: sykuo@cc.ee.ntu.edu.tw

Abstract

One of the important applications of software reliability growth models is to determine when to stop testing and release the software to the users. The Hyper–Geometric Distribution software reliability growth Model (HGDM) has been developed to estimate the number of faults initially resident in a program at the beginning of the test/debug phase. In this paper based on the HGDM, we investigate the cost–reliability optimal software release policy which not only minimizes the total software cost but also satisfies the software reliability requirement. The total software cost here includes the penalty cost which should be paid by the manufacturer if the software is delivered after the scheduled delivery time. The main result is that the optimal release time can be determined and shown to be finite. A numerical example is presented to illustrate the cost–reliability optimal software release policy.

1. Introduction

In recent years, computer systems have been widely applied to the monitor and control critical systems, for example, in air traffic control, nuclear reactors, real–time military applications, and hospital patient monitoring systems. Therefore, the breakdown of a computer system, caused by software faults, may result in tremendous damage for social life. Software reliability is one of the key issues in the software product development. A model for software reliability assessment during testing phase is called a Software Reliability Growth Model (SRGM). In the literature, many software reliability growth models have been developed [1–5].

In addition to fulfilling the requirement on software reliability, one of the important applications of SRGMs is to determine when to stop testing and release the software to the users. In general, the longer the software is tested, the higher the software reliability is. However, delays in software release will increase testing and other costs. Hence, it is desirable to determine the optimal release time. Such decision problem is called an optimal software release problem, and has been studied by [6–12]. In [6], Okumoto and Goel addressed the cost optimal software release policy minimizing the total expected software cost. Yamada and Osaki [10] introduced the cost–reliability optimal software release policy which minimizes the total expected cost and satisfies the software reliability requirement. However, the cost model they proposed excludes the penalty cost due to the delay from the scheduled delivery time. In fact, the delay by the test will incur additional cost and the late release for operational use will also lead to users' dissatisfaction. Hence, it is usually assumed that additional penalty cost should be paid if a software is released after the scheduled delivery time. Therefore, Kapur *et al.* [11] discussed the cost–reliability optimal release policies with scheduled software delivery time. The underlying software reliability growth models in their approach are based on the Non–Homogeneous Poisson process (NHPP).

The Hyper–Geometric Distribution software reliability growth Model (HGDM) was first proposed by Tohma *et al.* [13]. A series of studies on the HGDM have been made recently [14–20]. In this paper, we investigate the cost–reliability optimal release policy with scheduled delivery time which not only minimizes

[1] Yi–Ping Chang is with Department of Business Mathematics, Soochow University, Taipei, Taiwan, R.O.C.

the total software cost but also satisfies the software reliability requirement. Note that the software cost includes the additional penalty cost if a software is released after the scheduled delivery time. The underlying software reliability growth model in our approach is the HGDM with with exponential or logistic learning factor [19]. The overall organization of this paper is as follows. A brief review of the HGDM with exponential or logistic learning factor is given in Section 2. The software cost model including the penalty cost is proposed in Section 3. The cost–reliability optimal software release policy with scheduled delivery time based on the HGDM are discussed in Section 4. A Numerical example is presented for illustration in Section 5 followed by the conclusions in Section 6.

Notations

m	number of initial software faults in a program
t_i	the i^{th} test instance, $i = 1, 2, \ldots, n$ where i represents the order of application.
w_i	number of faults newly discovered or rediscovered by t_i
u_i	testing resource (the number of test items, the number of testers, or CPU time, etc.) performed in t_i
p_{LT}, a, b	constant parameters of the HGDM with exponential or logistic learning factor
C_i	cumulative number of software faults discovered by $t_1, t_2, \ldots, t_i$
I^*	optimal software release time
I	$\inf\{i \geq 1 : \Delta(i) < 0\}$
I_α	$\inf\{i \geq 1 : 1 - \prod_{j=1}^{i}(1-p_j) \geq \alpha\}$
c_1	expected cost of fixing a fault during testing
c_2	expected cost of fixing a fault during operation ($c_2 > c_1 > 0$)
c_3	expected cost of testing per unit time ($c_3 > 0$)
c_4, c_5	nonnegative real numbers
D	scheduled software delivery time ($D > 0$)
$C_D(i)$	penalty cost function due to the delay from the scheduled software release time
$CT(i)$	total expected cost with the penalty cost when the software is released at t_i
$C^*(i)$	$(c_3 + C_D(i) - C_D(i-1))/(m(c_2 - c_1))$
I_f	$\inf\{i \geq I :\ \delta(i+1) \leq C^*(i+1)\}$
$\delta(i)$	$p_i \prod_{j=1}^{i-1}(1-p_j),\ \ i = 2, 3, \cdots$
$\Delta(i)$	$p_{i+1} - p_i - p_{i+1}p_i,\ \ i = 1, 2, \cdots$
S	$\{D-1 \leq i \leq I-1 :\ \delta(i) > C^*(i)$ and $\delta(i+1) \leq C^*(i+1)\}$
α	prespecified software reliability ($0 < \alpha < 1$)

Assumptions

1 Faults which have been discovered upon the application of t_i don't have to be fixed before the following test instance t_{i+1} is applied.

2 During the removal of discovered faults, no new faults will be inserted.

3 The w_i faults discovered by t_i are those taken randomly out of the m initial faults.

4 The number of test items (the number of testers, CPU time) performed in a test instance is assumed to be the same for all test instances, i.e., u_i is assumed to be a constant for i=1,2...,n.

2. Review of HGDM

In this section, we briefly review the Hyper–Geometric Distribution software reliability growth Model (HGDM) [13–16]. The HGDM has been developed to estimate the number of remaining software faults after the test/debug phase. In general, a program is assumed to have m faults initially when the test/debug phase starts. The collection of test operations performed in a day or a week is called a *"test instance"*. The test/debug phase consists of consecutive applications of test instances. Test instances are denoted by t_i, $i = 1, 2, \ldots, n$ in accordance with the order of applying them. The *"sensitivity factor"*, w_i, represents how many faults are discovered during the application of test instance t_i. Each fault is classified into one of the two categories, newly discovered faults or rediscovered faults. Some of the faults detected by t_i may have been detected previously by the application of t_1 through t_{i-1} test instances. Therefore, the number of newly detected faults during the application of the i^{th} test instance is not necessarily equal to w_i.

Considering the application of t_i, let C_{i-1} be the number of faults already detected so far by $t_1, t_2, \ldots, t_{i-1}$ and N_i be the number of faults newly detected by t_i. Then, some of the w_i faults may be those that are already counted in C_{i-1}, and the remaining w_i faults account for the newly detected faults. With the assumption that new faults will not be inserted into the program while correction is being performed, the conditional probability $Prob(N_i = x_i \mid$

m, w_i, C_{i-1}) can be formulated as

$$Pr\left(N_i{=}x_i \,|\, m, w_i, C_{i-1}\right) = \frac{\binom{m-C_{i-1}}{x_i}\binom{C_{i-1}}{w_i-x_i}}{\binom{m}{w_i}}, \quad (1)$$

where $\max(0, w_i - C_{i-1}) \le x_i \le \min(w_i, m - C_{i-1})$ for all $i \ge 0$, $C_{i-1} = \sum_{k=1}^{i-1} x_k$, $C_0 = 0$, $x_0 = 0$ and x_k is an observed instance of N_k. The expected value of C_i denoted by EC_i is [13–16]

$$\begin{cases} EC_0 = 0, \\ EC_i = m[1 - \prod_{j=1}^{i}(1-p_j)], \quad i = 1, 2, \ldots, \end{cases} \quad (2)$$

where

$$p_i = w_i/m. \quad (3)$$

There are various functions for p_i presented in [13–16]. For example, $p_i = u_i(ai+b)$ is applied [13–14]. In this paper, since a test instance is a time unit of the release times, we have one additional assumption that u_i is a constant. Hou *et. al.* [19] has proposed two p_i functions when u_i is assumed to be a constant based on the exponential and the S–shaped learning curves, respectively. The first function is

$$p_i = p_{LT}(1 - e^{-ai}),\ a > 0,\ 0 < p_{LT} \le 1, \quad (4)$$

which is called the *"exponential learning factor"*. The second function is

$$p_i = p_{LT}\frac{1}{1 + be^{-ai}},\ a > 0,\ b > 0,\ 0 < p_{LT} \le 1, \quad (5)$$

which is called the *"logistic learning factor"*.

In the following sections, we will discuss the cost–reliability optimal software release policy with scheduled software delivery time. The underlying software reliability growth model is the HGDM with exponential or logistic learning factor.

3. Software Cost Model

Okumoto and Goel [6] discussed the software optimal release policy from the cost–benefit viewpoint. However, the cost model they proposed excludes the penalty cost due to the delay from the scheduled delivery time. In general, additional penalty cost should be paid if a software is released after the scheduled delivery time [7–8]. In this paper, the penalty cost function for the HGDM is

$$C_D(i) = \begin{cases} 0, & i < D; \\ c_4 + c_5 g(i - D), & i \ge D. \end{cases} \quad (6)$$

In addition, the longer the software is delayed delivering, the more the manufacturer should pay the penalty cost. Therefore, $g(i)$ is assumed to satisfy the following four conditions A1–A4.

A1: $g(0) = 0$.

A2: $g(i)$ is increasing in i.

A3: $g(i)$ is a convex function of i. That is, $g(i+1) + g(i-1) \ge 2g(i)$ for all $i \ge 1$.

A4: $C_D(i)$ is a convex function of i. That is, $c_5 g(1) \ge c4$.

Including the penalty cost, the total software cost for the HGDM is given by

$$CT(i) = c_1 EC_i + c_2(m - EC_i) + c_3 i + C_D(i), i = 0, 1, 2, \cdots. \quad (7)$$

4. Cost–Reliability Optimal Release Policy with Scheduled Delivery Time

There is a common measure [12] for describing software reliability: the ratio of the cumulative number of newly discovered faults to the number of initial fault. Based on the above measure, the ratio of the expected number of the cumulative number of newly discovered faults at the i^{th} test instance to the number of initial total faults, denoted by

$$R(i) = \frac{EC_i}{m}, \quad i = 0, 1, 2, \cdots, \quad (8)$$

is the measure of software reliability for the HGDM. Apparently, the larger the value of $R(i)$ is, the higher the software reliability is. In practice, it is reasonable to stop testing and release the software when the cumulative number of newly discovered faults is larger than a prescribed portion of initial faults [12]. The cost–reliability optimal release policy is to determine the optimal release time by minimizing the total expected cost subject to reaching the prescribed level α of $R(i)$. Therefore, the cost–reliability optimal release problem with scheduled delivery time based on the HGDM can be stated as:

$$\begin{cases} \text{minimize } CT(i) \\ \text{subject to } R(i) \ge \alpha, \quad 0 \le \alpha < 1. \end{cases}$$

From Eq.(2), we have

$$\begin{cases} R(0) = 0, \\ R(i) = 1 - \prod_{j=1}^{i}(1-p_j), \quad i = 1, 2, \cdots, \end{cases} \quad (9)$$

and then $R(i)$ is increasing in i. Define

$$I_\alpha = inf\{i : R(i) \geq \alpha\}. \quad (10)$$

From Eq.(2), we have

$$\begin{cases} CT(0) = c_2 m, \\ CT(i) = c_1 m + (c_2 - c_1)m\prod_{j=1}^{i}(1-p_j) + c_3 i + C_D(i), i = 1, \cdots \end{cases} \quad (11)$$

For convenience, let

$$C^*(i) = \frac{c_3 + C_D(i) - C_D(i-1)}{(c_2 - c_1)m}, \quad i = 1, 2, \cdots \quad (12)$$

and

$$\begin{cases} \delta(1) = p_1; \\ \delta(i) = p_i \prod_{j=1}^{i-1}(1-p_j), \quad i = 2, 3, \cdots. \end{cases} \quad (13)$$

Since

$$\begin{aligned} CT(1) - CT(0) &= c_3 - (c_2 - c_1)mp_1 + C_D(1) \\ &= (c_2 - c_1)m\{C^*(1) - \delta(1)\} \end{aligned}$$

and

$$CT(i+1) - CT(i) = (c_2 - c_1)m\{C^*(i+1) - \delta(i+1)\}, i = 1, 2, \cdots$$

we have the following lemma.

Lemma 1. For $i = 0, 1, 2, \cdots$, we have:

(i) if $C^*(i+1) > \delta(i+1)$, then $CT(i+1) > CT(i)$;

(ii) if $C^*(i+1) < \delta(i+1)$, then $CT(i+1) < CT(i)$;

(iii) if $C^*(i+1) = \delta(i+1)$, then $CT(i+1) = CT(i)$. □

Let

$$\Delta(i) = p_{i+1} - p_i - p_{i+1}p_i, \quad i = 1, 2, \cdots, \quad (14)$$

and then

$$\begin{cases} \delta(2) - \delta(1) = \Delta(1); \\ \delta(i+1) - \delta(i) = \{\prod_{j=1}^{i-1}(1-p_j)\}\Delta(i), i = 2, 3, \cdots \end{cases} \quad (15)$$

Define

$$I = inf\{i \geq 1 : \Delta(i) < 0\}. \quad (16)$$

For the exponential and the logistic learning factors, we have the following lemmas, respectively [17].

Lemma 2. If $p_i = p_{LT}(1 - e^{-ai})$, then:

(i) $\delta(i) \leq \delta(i+1)$ for $1 \leq i \leq I-1$;

(ii) $\delta(i) > \delta(i+1)$ for $i \geq I$. □

Lemma 3. If $p_i = p_{LT}/(1 + be^{-ai})$, then:

(i) $\Delta(i)$ is decreasing for $i \geq I$;

(ii) $\delta(i) \leq \delta(i+1)$ for $1 \leq i \leq I-1$ and $\delta(i) > \delta(i+1)$ for $i \geq I$. □

For convenience, define

$$I_f = \inf\{i \geq I : \ \delta(i+1) \leq C^*(i+1)\}, \quad (17)$$

and

$$S = \{D-1 \leq i \leq I-1 : \delta(i) > C^*(i) \text{ and } \delta(i+1) \leq C^*(i+1)\}. \quad (18)$$

We have the following Lemma [17].

Lemma 4. Suppose that $g(i)$ satisfies conditions A1–A4. If $p_i = p_{LT}(1 - e^{-ai})$ or $p_i = p_{LT}/(1 + be^{-ai})$, then I_f is finite and unique. □

To derive the *"cost-reliability"* optimal release time, the following theorem on the *"cost"* optimal release time I^* need to be obtained in advanced [17].

Theorem 1. Suppose that $g(i)$ satisfies conditions A1–A4. If $p_i = p_{LT}(1 - e^{-ai})$ or $p_i = p_{LT}/(1 + be^{-ai})$, we have:

(i) if $D > I$, then I^* satisfies $CT(I^*) = \min_{i \in \{0, I_f\}} CT(i)$;

(ii) if $D \leq I$, then I^* satisfies $CT(I^*) = \min_{i \in S \cup \{0, I_f\}} CT(i)$, where I_f and S are defined in Eq.(17) and Eq.(18). □

Note that Theorem 1 can be consolidated into $I^* \in S \cup \{0, I_f\}$. From the definitions of I_f and S, we have $I_f = \max\{S \cup \{0, I_f\}\}$. From the proof of Theorem 1, the following lemma can be easily obtained.

Lemma 5. If $p_i = p_{LT}(1 - e^{-ai})$ or $p_i = p_{LT}/(1 + be^{-ai})$, then $CT(i)$ is increasing in i $(i \geq I_f)$. □

Applying Theorem 1 (the theorem on cost optimal release time) and Lemma 5, we can obtain the following theorem on the cost–reliability optimal release time I^*.

Theorem 2. Suppose that $p_i = p_{LT}(1 - e^{-ai})$ or $p_i = p_{LT}/(1 + be^{-ai})$. If $I_\alpha \geq I_f$, then $I^* = I_\alpha$; otherwise, I^* satisfies $CT(I^*) = \min_{i \in W} CT(i)$, where

$W = [S \cup \{0, I_f\}] \cap \{I_\alpha, I_\alpha + 1, \cdots, I_f\}$ and S is defined in Eq.(18).

Proof: If $I_\alpha \geq I_f$, using Lemma 5 we have $I^* = I_\alpha$. If $I_\alpha < I_f$, using Lemma 5 we have $I^* \in \{I_\alpha, I_\alpha + 1, \cdots, I_f\}$. Furthermore, from Theorem 1, we have $I^* \in S \cup \{0, I_f\}$ and $I_f = \max\{S \cup \{0, I_f\}\}$ based on the cost criterion. Therefore, if $I_\alpha < I_f$, we have $I^* \in [S \cup \{0, I_f\}] \cap \{I_\alpha, I_\alpha + 1, \cdots, I_f\}$. □

In Theorem 2, we have proposed the procedure of determining I^* based on the HGDM with exponential or logistic learning factor. Furthermore, from Lemma 4 I^* can be shown to be finite.

5. Numerical Examples

In this section, a numerical example is used to illustrate the cost-reliability optimal release policy with scheduled delivery time based on the HGDM with exponential learning factor. Note that the following argument is the same for the HGDM with logistic learning factor.

The data set used in this analysis is the test/debug data of a software system [21]. Since the test data is reported per week, a test instance is "a week of observation". The cumulative number of discovered faults for the 24 test instances are gathered. Since the testing resource u_i is a constant, assumption (4) on Page 3 holds. The least squares estimates of the parameters of the HGDM with exponential learning factor are $\widehat{m} = 2300$, $\widehat{a} = 0.1206$, and $\widehat{p_{LT}} = 0.1602$ [19].

The parameters of $CT(i)$ are taken from [6]: $c_1 =$ 1$ per fault, $c_2 =$ 5$ per fault, $c_3 =$ 10$ per week, and $c_4 =$ 1$. The various values of c_5 in the penalty cost function $C_D(i)$ are assumed to be 1, 5, 10, 20, and 40, respectively, for comparison. Suppose the software reliability requirement is 0.95 (i.e., $\alpha = 0.95$), from Eq.(10) we have $I_\alpha = 25$. Furthermore, since the estimated number of initial faults is 2300 ($\widehat{m} = 2300$), from Eq.(8) reliability requirement is that at least 2185 faults should be discovered (i.e., no more than 115 faults remaining in the program). Considering the case that $g(i) = i^2$, the cost and the cost–reliability optimal release times are given in Table 1. Table 1 also indicates when c_5 increases, the cost–reliability optimal release time will become smaller. For example, with $c_5 = 1$ and $D = 10$, we have the cost–reliability optimal release time $I^* = 28$ where the total expected cost $CT(i)$ is minimized and the reliability requirement is also achieved. Since 24 test instances have been collected, the 25^{th}, 26^{th}, 27^{th} and 28^{th}test instances should be collected and tested by the test engineers before the software is released. With $c_5 = 5$ and $D = 10$, we have the cost–reliability optimal release time $I^* = 25$ where $CT(i)$ is not minimized but the reliability requirement is achieved. Therefore, the 25^{th} test instance should be collected and tested by the test engineers before the software is released. Note that $CT(i)$ is minimized at $i = 22$ and $R(22) = 0.919$ but the reliability requirement 0.95 is not achieved. The graphical interpretation is shown in Figure 1

6. Conclusions

In this paper, we have discussed the cost–reliability optimal release policy with scheduled delivery time based on the HGDM with exponential or logistic learning factor. The effect of the penalty cost on the cost–reliability optimal release policy was analyzed. It is concluded that earlier release of the software system is a cost–effective policy if the penalty cost is considered. In addition, the cost–reliability optimal release time I^* has been shown to be finite.

Table 1. Cost and cost-reliability optimal release times based on the HGDM with exponential learning factor ($g(i) = i^2$, $\alpha = 0.95$, $I_\alpha = 25$).

c_5	Scheduled delivery time: $D = 10$	
	Cost optimal release time I^* and $R(I^*)$	Cost–reliability optimal release time I^* and $R(I^*)$
1	28 , 0.970	28 , 0.970
5	22 , 0.919	25 , 0.951
10	19 , 0.970	25 , 0.951
20	16 , 0.794	25 , 0.951
40	14 , 0.725	25 , 0.951

c_5	Scheduled delivery time: $D = 20$	
	Cost optimal release time I^* and $R(I^*)$	Cost–reliability optimal release time I^* and $R(I^*)$
1	30 , 0.979	30 , 0.979
5	26 , 0.958	26 , 0.958
10	24 , 0.942	25 , 0.951
20	23 , 0.932	25 , 0.951
40	21 , 0.905	25 , 0.951

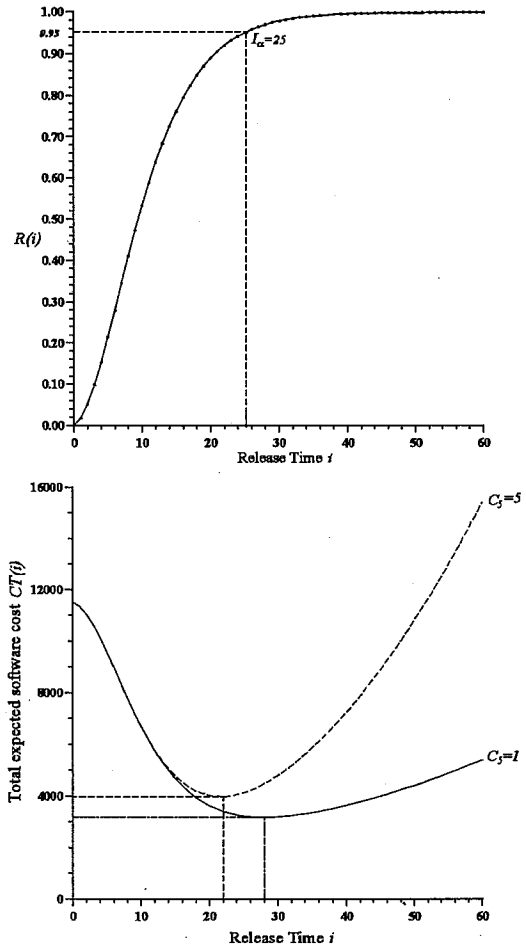

Figure 1. An illustration of the cost–reliability optimal release problem based on the HGDM with exponential learning factor ($D = 10$, $g(i) = i^2$, $\alpha = 0.95$, $I_\alpha = 25$).

Acknowledgment. We would like to express our gratitude for the support of the National Science Council, Taiwan, R.O.C., under Grants NSC85–2221–E002–015. Reviewers' comments are also highly appreciated.

References

[1] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability — Measurement, Prediction, Application*, McGraw–Hill, New York, 1987.

[2] M. Xie, *Software Reliability Modeling*, World Scientific Publishing, Singapore, 1991.

[3] C. V. Ramamoorthy and F. B. Bastani, "Software reliability — status and perspectives," *IEEE Trans. on Software Engineering*, vol. 8, No. 4, pp. 354–371, 1982.

[4] G. J. Schick and R. W. Wolverton, "An analysis of competing software reliability models," *IEEE Trans. on Software Engineering*, vol. 4, pp. 104–120, March 1978.

[5] M. Ohba, "software reliability analysis models," *IBM J. Res. Develop.*, Vol. 28, No. 4, pp. 428–443, July 1984.

[6] K. Okumoto and A. L. Goel, "Optimum Release Time for Software Systems Based on Reliability and Cost Criteria", *J. System Software*, Vol. 1, pp. 315–318, 1980.

[7] H. S. Koch and P. Kubat, " Optimal Release Time of Computer Software", *IEEE Trans. Software Engineering*, Vol. SE–9, No.3, pp. 323–327, 1983.

[8] S. Yamada, H. Narihisa, and S. Osaki, "Optimum Release Policies for a Software System with a scheduled Delivery Time", *Int. J. Systems Science*, Vol. 15, pp. 905–914, 1984.

[9] S. M. Ross, "Software Reliability: The Stopping Problem", *IEEE Trans. Software Engineering*, Vol. SE–11, No.12, pp. 1472–1476, 1985.

[10] S. Yamada and S. Osaki, "Cost–Reliability Optimal Release Policies for Software Systems", *IEEE Trans. Reliability*, Vol.34, No.5, pp. 422–424, 1985.

[11] P. K. Kapur and R. B. Garg, "Cost–reliability Optimum Release Policies for a Software System under Penalty Cost", *Int. J. Systems Science*, Vol. 20, pp. 2547–2562, 1989.

[12] M. Xie, "On the determination of optimum software release time," *Proceedings of the 2nd International Symposium on Software Reliability Engineering*, pp. 218–224, May 1991, Austin, Texas.

[13] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper–geometric distribution", *IEEE Trans. on Software Engineering*, vol. 15, No. 3, pp. 345–355, March 1989.

[14] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model", *IEEE Trans. on Software Engineering*, vol. SE–17, No. 5, pp. 483–489, May 1991.

[15] R. Jacoby and Y. Tohma, "Parameter value computation by least square method and evaluation of software availability and reliability at service–operation by the hyper–geometric distribution software reliability growth model (HGDM)", *Proc. 13th Int. Conf. Software Engineering*, pp. 226–237, 1991.

[16] T. Minohara and Y. Tohma, "Parameter estimation of hyper–geometric distribution software reliability growth model by genetic algorithms", *Proc. 6th Int. Symposium on Software Reliability Engineering*, pp. 324–329 , October 1995, Toulouse, France.

[17] R. H. Hou, S. Y. Kuo, and Y. P. Chang, " Optimal Release Times for Software Systems with Scheduled Delivery Time Based on the HGDM," *IEEE Trans. on Computers (accepted for publication).*

[18] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Optimal Release Policy for Hyper–Geometric Distribution Software Reliability Growth Model," *IEEE Trans. on Reliability (accepted for publication).*

[19] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Applying Various Learning Curves to Hyper–Geometric Distribution Software Reliability Growth Model," *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pp. 7–16, November 1994, Monterey, California.

[20] R. H. Hou, S. Y. Kuo, and Y. P. Chang, " Efficient Allocation of Testing Resources for Software Module Testing Based on the Hyper-Geometric Distribution Software Reliability Growth Model," *Proceedings of the 7th International Symposium on Software Reliability Engineering*, pp. 289–298, October 1996, White Plains, New York.

[21] A. L. Goel, *Software reliability modeling and estimation technique*, Final Technical Report RADC–TR–82–263, 1983.