A Study of Uncertainty in Software Cost and Its Impact on Optimal Software Release Time

Bo Yang, *Member*, *IEEE*, Huajun Hu, and Lixin Jia

Abstract—For a software development project, management often faces the dilemma of when to stop testing the software and release it for operation, which requires careful decision making as it has great impact on both software reliability and project cost. In most existing research on the optimal software release problem, the cost considered was the Expected Cost (EC) of the project. However, what concerns management is the Actual Cost (AC) of the project rather than the EC. Treatment (such as minimization) of the EC may not ensure the desired low level of the AC due to the uncertainty (variability) involved in the AC. In this paper, we study the uncertainty in software cost and its impact on optimal software release time in detail. The uncertainty is quantified by the variance of the AC and several risk functions. A risk-control approach to the optimal software release problem is proposed. New formulations of the problem which are extensions of current formulations are developed and solution procedures are established. Several examples are presented. Results reveal that it seems crucial to take into account the uncertainty in software cost in the optimal software release problem; otherwise, unsafe decisions may be reached which could be a false dawn to management.

Index Terms—Cost estimation, nonhomogeneous Poisson process (NHPP), reliability, software release, time estimation.

1 Introduction

Broadly speaking, a software development process consists of four phases: requirements and specification, design, coding, and testing. Software testing is a verification process for software quality and reliability improvement, which features failure observation phenomenon and fault removal activities. As the testing proceeds, latent software faults could be identified and removed, resulting in reliability growth of the software being tested. Software reliability models (SRMs) are commonly used to monitor the testing process and to measure and predict present and future reliability of the software.

In order to achieve a satisfactory reliability level, software will normally be tested for a rather long time before it can be released for operation. The testing phase is generally the most costly and time-consuming phase, in which approximately 40-50 percent of the total amount of software development resource is consumed [1]. In some cases, e.g., for safety-critical software, a certain high reliability level is often required by the customer and, consequently, the software has to be tested even longer until this reliability requirement is satisfied.

From management's point of view, however, it is desirable to start the sale of the software product as early as possible and hence begin to make profit. A delay in the release of the software product may result in a loss of market share, which finally leads to reduced economic benefits of the software product [2]. To resolve this dilemma, appropriate decision making needs to be made regarding when to stop testing the software and release it to the customer. This is normally referred to as *the optimal software release problem* in the literature, which has attracted a lot of attention and research since the early 1980s, see [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22].

In the literature, this problem was generally formulated as a (constrained) optimization problem in one of the following ways:

Formulation 1. (See, e.g., [13], [14], [16], [19].)

$$\text{Minimize } E[C(T)]. \tag{1}$$

Formulation 2. (See, e.g., [6], [9], [11], [18], [21].)

$$\text{Minimize } E\,[C(T)], \tag{2}$$

$$\text{Subject to } R(x|T) \geq R_0. \tag{3}$$

Formulation 3. (See, e.g., [8].)

$$\text{Maximize } R(x|T), \tag{4}$$

$$\text{Subject to } E[C(T)] \leq C_0. \tag{5}$$

In the above formulations, T is the release time of the software product and $C(T)$ is the cost incurred, which is a function of the release time. $E[C(T)]$ is the expectation of $C(T)$ and $R(x|T)$ is the reliability of the software product if it is released at time T. R_0 is the required reliability level

- *B. Yang is with the Department of Industrial Engineering, School of Mechatronics Engineering, University of Electronic Science and Technology of China, No. 4, Section 2, North Jian She Road, Chengdu, Sichuan, 610054, China. E-mail: yangbo@uestc.edu.cn.*
- *H. Hu is with the Department of Information Engineering, School of Electronics Engineering, University of Electronic Science and Technology of China, No. 4, Section 2, North Jian She Road, Chengdu, Sichuan, 610054, China. E-mail: huhuajun@uestc.edu.cn.*
- *L. Jia is with the School of Electrical Engineering, Xi'an Jiaotong University, No. 28, West Xian Ning Road, Xi'an, Shaanxi, 710049, China. E-mail: lxjia@mail.xjtu.edu.cn.*

Manuscript received 3 Oct. 2007; revised 31 Mar. 2008; accepted 5 June 2008; published online 19 June 2008.
Recommended for acceptance by R. Jeffery.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2007-10-0282.
Digital Object Identifier no. 10.1109/TSE.2008.47.

before the software can be released and C_0 is the allowed cost (budget) for the software project.

Formulations 1 and 2 stem from the idea of cost minimization, with Formulation 2 further considering the reliability requirement for the software product. Formulation 3 stems from the idea of reliability maximization with a cost (budget) constraint since, in some cases, e.g., for safety-critical software, reliability is the greatest concern.

However, it is important to take note that software cost, $C(T)$, is a random variable by nature due to various aspects of uncertainty involved in the testing and operational phases of the software product. For example, until release time T, the number of software faults that will be detected and removed is random; thus, the cost incurred by fault removal activities in the testing phase is random. Similarly, the cost incurred by fault removal activities in the operational phase is random too. Furthermore, the length of the software product's life cycle is random [9]. There are other aspects of uncertainty involved in software cost estimation as well [23], [24].

Since $C(T)$ is a random variable, a natural way to treat it is to consider its expectation, $E[C(T)]$. In Formulations 1 and 2, $E[C(T)]$ is used as the objective function and, in Formulation 3, it is used as a constraint.

However, it is known that, for a random variable, there is a certain level of variability involved in its realization values, which can be described by, e.g., its variance. The value of one realization of a random variable can be quite far from the value of its expectation, with a probability that can be obtained from the distribution of the random variable. Therefore, we argue that, in the optimal software release problem, even if the *Expected Cost* (EC), $E[C(T)]$, is minimized (see Formulations 1 and 2), management could not have sufficient confidence that the *Actual Cost* (AC) incurred in the software project, $C(T)$, will be near this minimum unless the probability that this happens is shown to be high enough [22].

From the law of large numbers, it is known that the treatment (such as minimization) of the expectation of a random variable makes sense for decision making only if the random variable can be realized for a large number of times. This situation can be true for some industrial applications. For example, in a queuing system, if by certain means the expected queuing time is minimized, then from a long-run perspective this is meaningful since the queuing time, which is a random variable, will be realized for many times (as long as the queuing system is in operation), and hence we can expect that the *average* queuing time is small.

Unfortunately, this situation is not applicable to software development projects. In reality, a software development project will not be repeated for many times and, when a project is completed, the next project to start may be different from the previous one in many aspects. Thus, the *average* (expected) cost could not be manifested in a single software project or across software projects.

From the above discussions, it seems that, in the optimal software release problem, besides the expectation of software cost, one should also consider the uncertainty involved, i.e., the variability of software cost; otherwise, unsafe decisions could be made. Current formulations (1)-(5) could lead to the best point estimate of the optimal software release time; nevertheless, if the uncertainty (variability) of software cost is taken into account, a more informed and more appropriate decision could be made.

In this paper, we study the uncertainty in software cost and its impact on the optimal software release problem in detail. Our discussion will be based on nonhomogeneous Poisson process (NHPP) SRMs, for which the software failure process is assumed to follow an NHPP. Denoting by $N(t)$ the cumulative number of observed failures until time t, $N(t)$ follows Poisson distribution with parameter $m(t)$, i.e.,

$$\Pr\{N(t)=n\}=\frac{[m(t)]^n}{n!}e^{-m(t)},\quad n=0,1,2,\cdots, \tag{6}$$

where $m(t)\equiv E[N(t)]$ is the *mean value function* of the NHPP. The *failure intensity function*, $\lambda(t)$, is mathematically the derivative of the mean value function, i.e.,

$$\lambda(t)\equiv\frac{d}{at}m(t). \tag{7}$$

NHPP SRMs form a major class of SRMs. They have many good properties that account for their popularity among researchers and practitioners. For example, the calculation of the expected number of failures up to time t is very simple due to the existence of mean value function and the model parameters can be easily estimated using the Maximum Likelihood Estimation (MLE) method [2], [25], [26]. As a result, NHPP SRMs have been thoroughly studied in the literature and widely used in practice.

The remainder of this paper is organized as follows: In Section 2, we give a brief review on related research work. In Section 3, we study the uncertainty in software cost from the perspective of the AC's variance. In Section 4, we study the uncertainty from a risk perspective and we conduct detailed risk analyses for two representative software cost models. In Section 5, a new approach to the optimal software release problem based on the principle of risk control is proposed and solution procedures are established. Several examples are given to demonstrate the ideas behind and the solution procedures. Finally, some concluding remarks are given in Section 6.

2 Related Work

Since some early work on the optimal software release problem [3], [4], [5], [6], [7], new software cost models have been developed and used in related research in recent years. Pham [9] developed a software cost model with imperfect debugging, random life cycle, and penalty cost for the delay in software release. The idea of penalty cost was also suggested in [4], [10], and imperfect debugging was also addressed in [13]. Pham and Zhang [11] developed a generalized cost model which considered fault removal cost, warranty cost, and software risk cost due to software failures. The risk cost was also addressed in [17], [20]. Kimura et al. [12] developed a software cost model considering software maintenance cost during the warranty

period. Pham [15] presented a good review on advances in NHPP SRMs and software cost modeling.

To address testing effort and efficiency in software reliability modeling, Huang and Lyu [18], [19] proposed an SRM which incorporated a generalized logistic testing-effort function and developed a software cost model considering testing effort and efficiency.

When software testing is considered to be not continuous in time but to be constituted by several stages, with each stage terminated at the observation of a software failure, the optimal software release problem can be formulated as a sequential decision problem. The decision making involved becomes whether or not to enter the next testing stage at the end of current testing stage. Morali and Soyer [14] studied this problem by taking a Bayesian decision theoretic approach. Chang and Hung [17] studied this problem by formulating it as an open-loop-feedback-optimal control problem. Liu and Chang [20] developed a software cost model based on a non-Gaussian Kalman filter model.

Much of the existing research on the optimal software release problem is based on NHPP SRMs; however, there is also research on this topic based on other types of SRMs. Hou et al. [10], Boland and Singh [16], and Bolanda and Chuiv [21] studied the optimal software release problem based on the hypergeometric software reliability growth model, the Moranda geometric deeutrophication model, and a generalized Jelinski-Moranda model, respectively. In fact, the optimal software release problem is independent of the underlying SRMs used and it can be formulated and studied under any SRM.

It appears that existing research on the optimal software release problem has been focusing on developing more realistic software cost models. This could make the decision made by management more appropriate. Nevertheless, another important issue, i.e., that the formulations adopted for the problem, Formulations 1-3, do not take sufficient account of the uncertainty involved in software cost, has not been addressed in most of existing research except [22]. The deficiency in current research on optimal software release problem motivates our research presented in this paper.

3 Uncertainty in Software Cost

The AC of a software project, $C(T)$, is a random variable; thus, there exists a certain level of uncertainty (variability) of its realization values. There are two ways to quantify the uncertainty. One is by the variance of $C(T)$, the other is by the probability of an event of interest to management such as the probability that the AC will exceed the EC by 10 percent. The second way is a more accurate quantification of the uncertainty in $C(T)$.

In this section, we study the uncertainty in $C(T)$ in the first way, i.e., quantifying the uncertainty by the variance. In Section 4, we will study the uncertainty in the second way.

3.1 Software Cost Modeling

In general, software cost, $C(T)$, can be formulated as

$$C(T) = c_0 + \sum_{i=1}^{5} C_i(T). \tag{8}$$

In the above, c_0 is the setup cost (or called the initial testing cost [12]) for software testing [11], which is often assumed to be a constant. $C_1(T)$ is the cost incurred by fault removal activities in the testing phase, $C_2(T)$ is the cost incurred by fault removal activities in the operational phase, and $C_3(T)$ is the general cost of testing (e.g., payment to the testing team members); see, e.g., [2], [15]. $C_4(T)$ is the risk cost due to software failures [11], [17], [20] and $C_5(T)$ is the penalty cost for the delay in software release [4], [9], [10]. In existing research, different formulations of cost components $C_i(T)$, $1 \le i \le 5$, have been proposed. Moreover, other cost components can be considered and added to the cost model (8) as well.

It is known that the *costs of quality* can be categorized into *prevention costs*, *appraisal costs*, *internal failure costs*, and *external failure costs* [27]. Slaughter et al. [28] expatiated that, in software development, prevention costs include the costs of training staff in design methodologies, quality improvement meetings, software design reviews, etc.; appraisal costs include the costs of code inspections, testing, software measurement activities, etc.; internal failure costs include the costs of rework in programming, reinspection, retesting, etc.; external failure costs include field service and support, maintenance, liability damages, litigation expenses, etc. In the cost model (8), $C_1(T)$ and $C_3(T)$ can be viewed as mixtures of appraisal costs and internal failure costs; $C_2(T)$, $C_4(T)$, and $C_5(T)$ can be viewed as external failure costs. Therefore, it can be noted that the cost defined in (8), $C(T)$, is only part of the total cost incurred in the life cycle of the software product. In the literature, $C(T)$ is often called a software cost model, see, e.g., [6], [11], [12], [15]; however, it is important to keep in mind that the cost considered here is different from the total (life cycle) cost of a software product.

For the formulation of $C_1(T)$, generally, it is considered to be proportional to the number of software faults removed during the testing phase. Similarly, $C_2(T)$ is considered to be proportional to the number of software faults removed during the operational phase. Thus,

$$C_1(T) = c_1 N(T), C_2(T) = c_2[N(T_{LC}) - N(T)], \tag{9}$$

where T_{LC} is the life-cycle length of the software product. c_1 is the cost of removing a fault in the testing phase and c_2 is the cost of removing a fault in the operational phase; normally, we have $c_2 > c_1$. $C_3(T)$ is assumed to be a power function of testing time T [11], i.e.,

$$C_3(T) = c_3 T^{\kappa}, \tag{10}$$

where c_3 is a constant. The parameter κ $(0 < \kappa \le 1)$ reflects the fact that the increasing gradient is different in the beginning and at the end of testing [11]. In the simplest case, $\kappa = 1$ (see, e.g., [2], [9], [16], [21]).

For SRMs considering testing effort (see, e.g., [18], [19]), $C_3(T)$ is formulated as

$$C_3(T) = c_w[W(T)]^{\kappa}, \tag{11}$$

where $W(T)$ is the total testing-effort spent in $(0, T]$ and c_w is the cost per unit testing-effort expenditure.

The risk cost due to software failures, $C_4(T)$, is generally formulated as [11], [20]:

$$C_4(T) = c_4[1 - R(x|T)], \tag{12}$$

where c_4 is a constant and $R(x|T)$ is the reliability of the software if it is released at time T.

The penalty cost, $C_5(T)$, can be formulated as [4], [9], [10]

$$C_5(T) = I(T - T_d)C_p(T - T_d), \tag{13}$$

where T_d is the scheduled release time of the software, $I(\cdot)$ is an indicator function, which is defined as

$$I(t) \equiv \begin{cases} 1, & \text{if } t \geq 0 \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

and $C_p(\cdot)$ is the penalty cost function.

Despite the fact that many new software cost models have been proposed, for most of them, $C_1(T)$, $C_2(T)$, and $C_3(T)$ are common cost components that have been adopted. In fact, (8) is a generalization of a widely used basic software cost model (see, e.g., [2], [3], [16], [18], [19], [21]):

$$C(T) = C_1(T) + C_2(T) + C_3(T), \tag{15}$$

where $C_1(T)$, $C_2(T)$, and $C_3(T)$ have the same meaning as those in (8).

3.2 Uncertainty in Software Cost Measured by the Variance

From the software cost model (8), we can derive the variance of $C(T)$. Note that C_0, $C_3(T)$, $C_4(T)$, and $C_5(T)$ are not random variables, as can be seen from (10)-(13); thus, we have

$$Var[C(T)] = Var[C_1(T) + C_2(T)]. \tag{16}$$

For NHPP SRMs, for any fixed values of T and T_{LC}, both $N(T)$ and $N(T_{LC}) - N(T)$ are Poisson distributed random variables, with parameters $m(T)$ and $m(T_{LC}) - m(T)$, respectively. Furthermore, because NHPP has independent increments, $N(T)$ and $N(T_{LC}) - N(T)$ are independent. Therefore, for NHPP SRMs, the variance given by (16) becomes

$$\begin{aligned} Var[C(T)] &= c_1^2 Var[N(T)] + c_2^2 Var[N(T_{LC}) - N(T)] \\ &= c_1^2 m(T) + c_2^2[m(T_{LC}) - m(T)]. \end{aligned} \tag{17}$$

In some related research, the following assumptions are made:

1. The life cycle of the software product is infinity.
2. $m_\infty \equiv \lim_{t\to\infty} m(t)$ exists.

These assumptions are not very restrictive and could be valid under some real-life situations. Assumption 1 is made by the fact that, generally speaking, a software product will be used much longer than the time spent in its testing phase; thus, the life cycle of the software product could be assumed to be infinity [2]. Assumption 2 can be satisfied by NHPP models with finite failures, e.g., the Goel-Okumoto (G-O) model [29] and the S-shaped model [30], [31]. Therefore, we also make these two assumptions in our research. The variance of $C(T)$ for NHPP SRMs, given by (17), thus becomes

$$Var[C(T)] = c_2^2 m_\infty - (c_2^2 - c_1^2)m(T). \tag{18}$$

TABLE 1
Model Parameters Used in [32]

Cost Parameters	SRM Parameters	Reliability Requirement
$c_1 = 200$,	$\hat{a} = 33.99$,	$x = 100$,
$c_2 = 1500$, $c_3 = 10$	$\hat{b} = 0.00579$	$R_0 = 0.8$

Because $c_2 > c_1$ and $m(\cdot)$ is a monotonous increasing function with $m(0) = 0$, it can be obtained from (18) that $Var[C(T)]$ is a monotonous decreasing function with T and

$$\begin{aligned} Var[C(T)]_{\max} &= c_2^2 m_\infty \quad (T = 0), \\ Var[C(T)]_{\min} &= c_1^2 m_\infty \quad (T = \infty). \end{aligned} \tag{19}$$

Very often $c_1^2 m_\infty$ takes a rather big value; thus, the variance of the AC would not be small, i.e., a certain level of uncertainty is involved in software cost which may not be negligible.

To exemplify this, we give an example here.

Example 1. Consider the optimal software release problem studied in [32]. It took Formulation 2 and the SRM used was the G-O model whose mean value function is

$$m(t) = a(1 - e^{-bt}), \tag{20}$$

where a and b are constants. The G-O model is one of the earliest NHPP SRMs developed and has far-reaching influence on later software reliability modeling; thus, it has been widely used in the literature. For the G-O model, we have

$$m_\infty = a. \tag{21}$$

In [32], the basic software cost model given by (15) was used, with $C_3(T)$ taking formulation (10), and $\kappa = 1$. Table 1 summarizes the model parameters used in [32]. The SRM parameters were estimated from the real software project data, reported in [29], which consists of 26 time-between-failure data obtained in the testing phase.

The optimal software release time was obtained as $T^* = 774$, under which the EC is $E[C(T^*)] = 1.50 \times 10^4$.

In the original work [32], no consideration of the uncertainty in $C(T)$ was given. It would be interesting to study how much the uncertainty is and thus examine the validity of the optimal software release time obtained.

The variance of $C(T^*)$ can be obtained from (18), which is $Var[C(T^*)] = 2.21 \times 10^6$.

This result implies that, on average, the AC deviates from the EC by 1.49×10^3, which is approximately 9.9 percent of the minimum EC.

4 Risk Analysis in Optimal Software Release Problem

4.1 Risk Functions

Besides the variance of $C(T)$, the uncertainty in software cost can be quantified in another way. Denote by $P_1(T)$ the probability that the AC of a software project is higher than the EC, i.e.,

$$P_1(T) \equiv \Pr\{C(T) > E[C(T)]\}. \tag{22}$$

$P_1(T)$ is a function of software release time T. This probability can be viewed as a kind of *risk*, which is of great concern to management. If this risk is controlled to a reasonable level, then, from (22), it can be seen that, when the EC is minimized, the AC will have little chance to become higher than this minimum, which is desirable.

A more generalized risk function can be defined as follows:

$$P_2^{\alpha}(T) \equiv \Pr\{C(T) > (1+\alpha)\cdot E[C(T)]\}, \tag{23}$$

where α $(\alpha \geq 0)$ is a constant representing the allowed margin for the AC being higher than the EC. $P_1(T)$ is in fact a special case of $P_2^{\alpha}(T)$ for which $\alpha = 0$.

4.2 Risk Analysis for the Basic Software Cost Model

For any software cost model used in the optimal software release problem, it is essential to conduct analysis of the risk involved, i.e., $P_1(T)$ or $P_2^{\alpha}(T)$. In this paper, our research will focus on two representative software cost models. In Section 4.2, we conduct risk analysis for the basic software cost model (15) and, in Section 4.3, we conduct risk analysis for a generalized cost model proposed in [11]. Risk analysis for other software cost models can be carried out in a similar fashion.

Take formulation (10) for $C_3(T)$, then, under Assumptions 1 and 2, the basic software cost model (15) becomes

$$C(T) = c_1 N(T) + c_2[N(\infty) - N(T)] + c_3 T^{\kappa} \tag{24}$$

and the EC becomes

$$E[C(T)] = c_1 m(T) + c_2[m(\infty) - m(T)] + c_3 T^{\kappa}. \tag{25}$$

The cost models (24)-(25) have been widely used in existing research, see, e.g., [2], [3], [16], [21]. For the basic software cost model (24), we have the following results:

Theorem 1. *If the software testing process is modeled by an NHPP SRM, then, under Assumptions 1 and 2, $P_1(T)$ for the basic software cost model (24) is*

$$P_1(T) = 1 - e^{-m_{\infty}} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_{\infty} - m(T)]^j}{j!} \tag{26}$$

and its limit is

$$\lim_{T\to\infty} P_1(T) = 1 - e^{-m_{\infty}} \sum_{k=0}^{\langle m_{\infty}\rangle} \frac{(m_{\infty})^k}{k!}. \tag{27}$$

In the above,

$$k_{up} \equiv \langle \gamma_1 \rangle, \quad j_k \equiv \left\langle \frac{c_1(\gamma_1 - k)}{c_2} \right\rangle, \qquad \gamma_1 \equiv \frac{c_1 m(T) + c_2[m_{\infty} - m(T)]}{c_1}. \tag{28}$$

The notation $\langle x \rangle$ represents the largest integer value that is less than or equal to x. The proof is similar to that given in [22] and thus is omitted here.

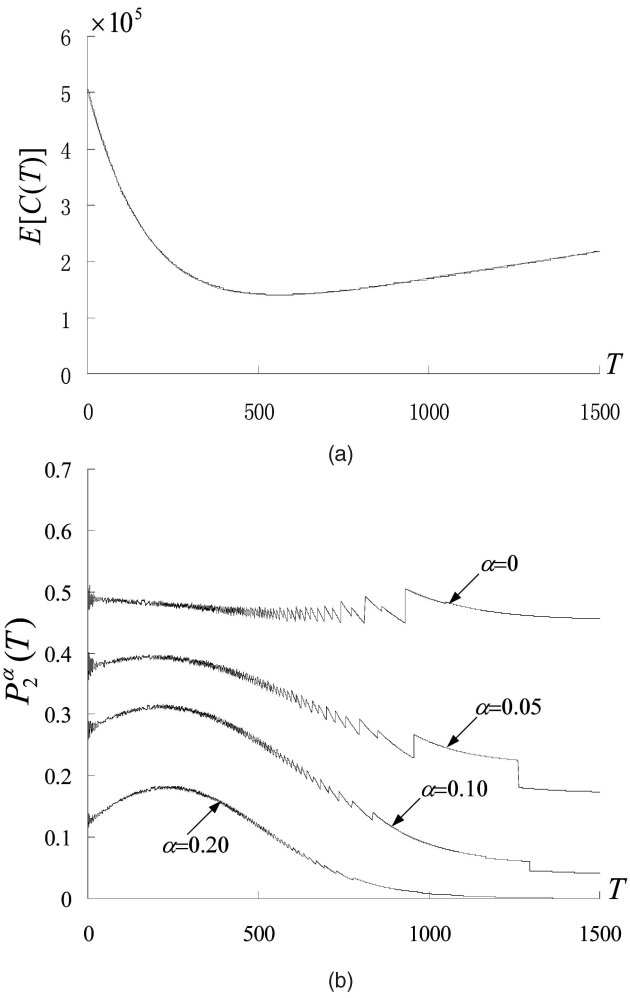

Fig. 1. Plots of (a) $E[C(T)]$ and (b) $P_2^{\alpha}(T)$ for the basic software cost model (24), using model parameters in Table 1.

Theorem 2. *If the software testing process is modeled by an NHPP SRM, then, under Assumptions 1 and 2, $P_2^{\alpha}(T)$ for the basic software cost model (24) is*

$$P_2^{\alpha}(T) = 1 - e^{-m_{\infty}} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_{\infty} - m(T)]^j}{j!}, \tag{29}$$

where

$$k_{up} \equiv \langle \gamma_2 \rangle, \quad j_k \equiv \left\langle \frac{c_1(\gamma_2 - k)}{c_2} \right\rangle, \qquad \gamma_2 \equiv \frac{(1+\alpha)c_1 m(T) + (1+\alpha)c_2[m_{\infty} - m(T)] + \alpha c_3 T^{\kappa}}{c_1}. \tag{30}$$

The proof is given in the Appendix.

Theorems 1 and 2 provide close-form formulations to calculate $P_1(T)$ and $P_2^{\alpha}(T)$ for the basic software cost model (24). Here, we give two illustrative examples.

Example 2. Reconsider Example 1. The plots of $E[C(T)]$ and $P_2^{\alpha}(T)$ with different values of α versus T are shown in Figs. 1a and 1b, respectively. Note that the curve in Fig. 1b, for which $\alpha = 0$, is in fact the plot of $P_1(T)$.

Since $T^* = 774$, we have $P_1(T^*) = 0.47$ and $P_2^{0.1}(T^*) = 0.15$, obtained from (26) and (29), respectively. From (27), the limit of $P_1(T)$ is obtained as $P_1(\infty) = 0.45$, as can be seen in Fig. 1b. These results mean that if we take the optimal software release time as 774, then, although the EC of the software project is nearly minimized, there is a risk (probability) of 0.47 that the AC incurred will exceed the minimized EC and there is a risk of 0.15 that the AC incurred will exceed the minimized EC by 10 percent.

Example 3. It is interesting to study the properties of $P_1(T)$ and $P_2^{\alpha}(T)$ for the basic software cost model (24) under other model parameter values. Here, we take Formulation 1 for the optimal software release problem and we take model parameter values as those used in [11].

The cost parameters were estimated from real project data collected by some AT&T researchers [11], which are $c_1 = 60$, $c_2 = 3{,}600$, and $c_3 = 700$.

In [11], the SRM used was the G-O model. From the failure data recorded in a real software project (see Table 1 in [11]), model parameters were estimated as $\hat{a} = 142.32$ and $\hat{b} = 0.1246$.

Under these parameters, the optimal software release time is obtained as $T^* = 36$, under which the minimum EC is $E[C(T^*)] = 39{,}417$.

The plots of $E[C(T)]$ and $P_2^{\alpha}(T)$ with different values α versus T are shown in Figs. 2a and 2b, respectively. It can be seen that the risk values at the optimal release time are $P_1(T^*) = 0.47$ and $P_2^{0.1}(T^*) = 0.21$.

From Examples 2 and 3, it can be seen that, by the analyses of $P_1(T)$ and/or $P_2^{\alpha}(T)$ together with the EC, a more reasonable optimal software release time could be obtained. For example, in Example 3, it can be seen in Fig. 2 that if we increase the release time, e.g., take $T' = 52$, then $P_1(T')$ will reach its minimum at 0.30 and $P_2^{0.1}(T') = 0.02$, i.e., the risks of $P_1(T')$ and $P_2^{0.1}(T')$ will reduce by 36.2 percent and 90.5 percent, respectively. The EC will increase by 16.0 percent though, i.e., $E[C(T')] = 45{,}713$. From a risk-control perspective, T' may be a better alternative for software release time than T^*.

If the increase of software release time from 36 to 52 is considered to be unacceptable, e.g., under the pressure of scheduled software delivery, then management could make a compromise between the reduction of the risks and the increase of the release time. For example, management could take $T' = 38$, under which $P_1(T') = 0.39$, $P_2^{0.1}(T') = 0.14$, and $E[C(T')] = 39{,}989$. It can be seen that, even by a small increase of the release time (from 36 to 38), the risks can be reduced to some extent.

Obviously, either taking T^* or T' as the optimal software release time is essentially a matter of trade-off, which is at the discretion of management. In Section 5, we will present a risk-control approach to the optimal software release problem.

In Figs. 1b and 2b, we can also observe the patterns of $P_1(T)$ and $P_2^{\alpha}(T)$ for the basic software cost model (24). The curve of $P_1(T)$ has a bit of fluctuation when T is small; then there is a notch when T increases, after which the curve converges to the value given by (27) when T approaches infinity. The fluctuation of the curve is caused by the operation of $\langle\cdot\rangle$ in (28). The curve of $P_2^{\alpha}(T)$, $\alpha > 0$ is roughly convex, forming a bulge, and approaches zero when T approaches infinity.

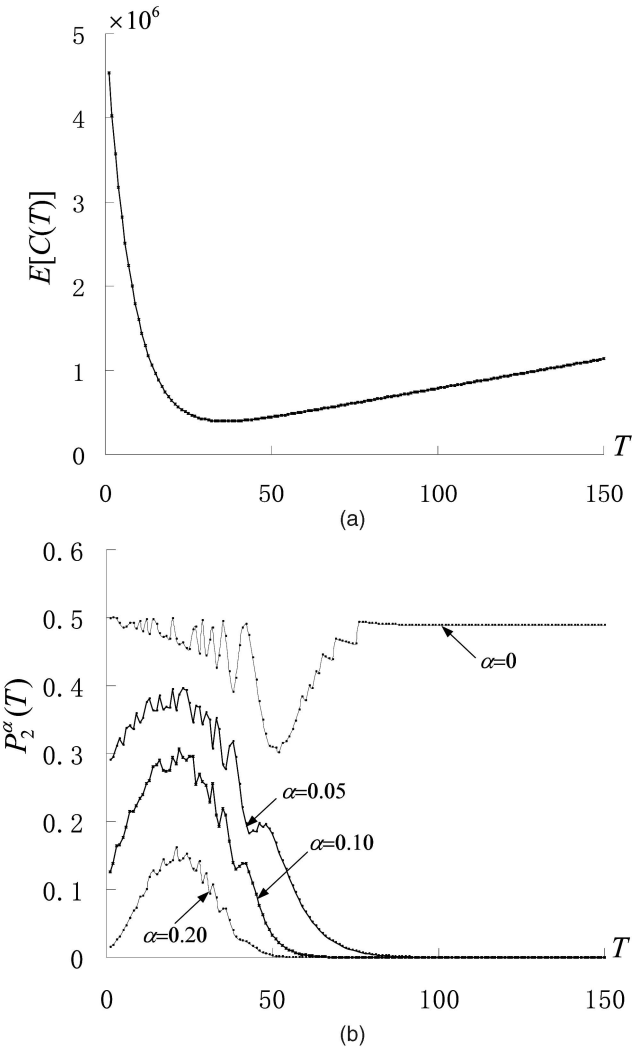

Fig. 2. Plots of (a) $E[C(T)]$ and (b) $P_2^{\alpha}(T)$ for the basic software cost model (24), using model parameter values as those used in [11].

4.3 Risk Analysis for the Generalized Software Cost Model

In this section, we conduct risk analysis on a more complicated cost model, the generalized software cost model proposed in [11], which is

$$C(T) = c_0 + c_1 \sum_{i=1}^{N(T)} Y_i + c_2 \sum_{i=N(T)}^{N(T+T_w)} W_i + c_3 T^{\kappa} + c_4[1 - R(x|T)], \tag{31}$$

where $N(t)$ is the cumulative number of observed software failures until time t, $\{N(t), t \geq 0\}$ is an NHPP with a mean value function of $m(t)$, T_w is the warranty time, and Y_i and W_i are times spent to remove the ith software fault in the testing phase and in the operational phase, respectively. Y_is are assumed to be independent and identically distributed (i.i.d.) random variables, each following truncated exponential

distribution with parameters λ_y and T_{0y}; W_is are also assumed to be i.i.d. random variables, each following truncated exponential distribution with parameters λ_w and T_{0w} [11]. This generalized software cost model seems to be the first one that considered both the risk cost and the warranty cost and has a strong influence on later software cost modeling.

Take the expectation on both sides of (31), then the EC is obtained as follows [11]:

$$E[C(T)] = c_0 + c_1\mu_y m(T) + c_2\mu_w[m(T+T_w) - m(T)] + c_3T^\kappa + C_4[1 - R(x|T)], \tag{32}$$

where $\mu_y = E[Y_i]$ and $\mu_w = E[W_i]$, which can be calculated by

$$\mu_y = \frac{1-(\lambda_y T_{0y}+1)e^{-\lambda_y T_{0y}}}{\lambda_y(1-e^{-\lambda_y T_{0y}})}, \quad \mu_w = \frac{1-(\lambda_w T_{0w}+1)e^{-\lambda_w T_{0w}}}{\lambda_w(1-e^{-\lambda_w T_{0w}})}. \tag{33}$$

For the generalized software cost model (31), the risk function $P_1(T)$ defined by (22) becomes

$$P_1(T) = \Pr\left\{ c_1 \sum_{i=1}^{N(T)} Y_i + c_2 \sum_{i=N(T)}^{N(T+T_w)} W_i > s_1 \right\}, \tag{34}$$

where

$$s_1 \equiv c_1\mu_y m(T) + c_2\mu_w[m(T+T_w) - m(T)]. \tag{35}$$

The risk function $P_2^\alpha(T)$ defined by (23) becomes

$$P_2^\alpha(T) = \Pr\left\{ c_1 \sum_{i=1}^{N(T)} Y_i + c_2 \sum_{i=N(T)}^{N(T+T_w)} W_i > s_2 \right\}, \tag{36}$$

where

$$s_2 \equiv (1+\alpha)s_1 + \alpha\{c_0 + c_3T^\kappa + c_4[1 - R(x|T)]\}. \tag{37}$$

In the optimal software release problem, the evaluation of software reliability is of major importance. Software reliability measurement is often used as a constraint in the optimization problem, see (3), and, in some cases, the cost model, (2), itself involves the evaluation of software reliability, see, e.g., (31). In the original work [11], the software reliability measurement used was the *testing reliability*, i.e.,

$$R(x|T) = e^{-[m(T+x)-m(T)]}. \tag{38}$$

The testing reliability measurement has been used in most related research, see, e.g., [12], [15], [18].

However, from the customer's point of view, the *operational reliability* is a more meaningful and appropriate measurement of software reliability [32]. Therefore, in our research, we adopt the operational reliability for software reliability measurement, i.e.,

$$R(x|T) = e^{-\lambda(T)\cdot x}, \tag{39}$$

where $\lambda(\cdot)$ is the failure intensity function given by (7).

For the generalized software cost model (31), due to the model complexity, analytical results for calculating $P_1(T)$ and $P_2^\alpha(T)$ such as (26) and (29) are very difficult to derive.

TABLE 2
Model Parameters Used in [11]

Cost Parameters	SRM Parameters	Other Parameters
$c_0 = 50$, $c_1 = 60$, $c_2 = 3600$, $c_3 = 700$, $c_4 = 50000$	$\hat{a} = 142.32$, $\hat{b} = 0.1246$	$x = 0.05$, $\kappa = 0.95$, $T_w = 20$, $\mu_y = 0.1$, $\mu_w = 0.5$

In this case, Monte Carlo simulation can be used to obtain $P_1(T)$ and $P_2^\alpha(T)$.

For a given value of T, the simulation steps to calculate $P_1(T)$ are listed as follows:

1. Generate a value of n, which is a realization of a Poisson distributed random variable N_1 with mean $m(T)$. Generate a value of m, which is a realization of a Poisson distributed random variable N_2 with mean $m(T+T_w) - m(T_w)$. N_1 and N_2 are independent.
2. Generate $y_1, \cdots, y_n$, which are n independent realizations of a random variable that follows truncated exponential distribution with parameters λ_y and T_{0y}.
3. Generate $w_1, \cdots, w_m$, which are m independent realizations of a random variable that follows truncated exponential distribution with parameters λ_w and T_{0w}.
4. Calculate $z \equiv c_1 \sum_{i=1}^{n} y_i + c_2 \sum_{i=1}^{m} w_i$. Calculate s_1 defined by (35).
5. Repeat Steps 1-4 for l times. Denote by l_0 the number of times that $z > s_1$ happens and l_0/l is an approximation of $P_1(T)$.

The calculation of $P_2^\alpha(T)$ can be carried out in a similar fashion.

Example 4. To illustrate, we consider the example presented in [11]. It took Formulation 1 and the cost model used was the generalized software cost model (31). Table 2 summarizes the model parameters [11].

For parameters not given in [11], we set

$$T_{0y} = 100, \quad T_{0w} = 500, \quad \lambda_y = 2.0, \quad \lambda_w = 2.0, \tag{40}$$

under which the values of μ_y and μ_w, obtained from (33), become the same as those in Table 2. We carry out $l = 5{,}000$ iterations for any fixed value of T.

The plots of $E[C(T)]$ and $P_2^\alpha(T)$ with different values of α versus T are shown in Figs. 3a and 3b, respectively. Under Formulation 1, the optimal software release time was obtained as $T^* = 33$ [11], under which $E[C(T^*)] = 24{,}854$. However, in Fig. 3b, it can be seen that the risks under the optimal release time are $P_1(T^*) = 0.82$ and $P_2^{0.1}(T^*) = 0.49$. These values are too large to be acceptable.

This example again demonstrates that, in the optimal software release problem, the uncertainty in software cost should not be ignored. If it is ignored, then the optimal software release time obtained, such as $T^* = 33$ in this example, would give management a false dawn that the cost of this software project would be at a low level (e.g., has been minimized). In fact, it may be possible (with a high probability of 0.82 in this example) that the AC of the software project would exceed this level. As a result,

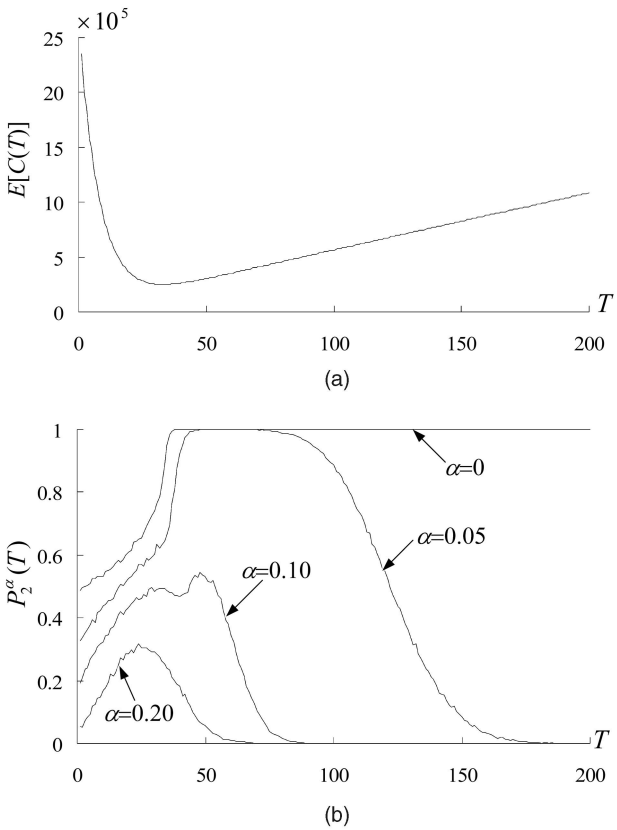

Fig. 3. Plots of (a) $E[C(T)]$ and (b) $P_2^{\alpha}(T)$ for the generalized software cost model (31), using model parameters in Table 2 and (40).

management may be deceived and no proper actions would be taken in a timely manner to prevent the project from running over budget.

In Fig. 3b, we can observe the patterns of $P_1(T)$ and $P_2^{\alpha}(T)$ for the generalized software cost model (31). The curve for which $\alpha = 0$ in Fig. 3b has a similar pattern to that in Fig. 2b, with the former having only part of the latter (from the value of T that minimizes $P_1(T)$ in Fig. 2b onward). The pattern of $P_2^{\alpha}(T)$ for the generalized software cost model (31) is similar to that for the basic software cost model (24), i.e., the curves are roughly convex and approach zero when T approaches infinity.

5 A Risk-Control Approach to Optimal Software Release Problem

From discussions in Section 4, it is clear that in the optimal software release problem the uncertainty involved in software cost should not be neglected since it may incur great risk which, if not considered, could make the software project run over budget. In this section, we propose a risk-control approach to the optimal software release problem and we develop new problem formulations that are extensions of current formulations.

5.1 Minimization of Expected Cost with Risk Constraint

To address the uncertainty in software cost and control the risk incurred, a risk constraint can be added to Formulations 1 and 2 of the optimal software release problem.

Formulation 4.

$$\text{Minimize } E[C(T)], \tag{41}$$

$$\text{Subject to } P_2^{\alpha}(T) \le \beta. \tag{42}$$

Formulation 5.

$$\text{Minimize } E[C(T)], \tag{43}$$

$$\text{Subject to } R(x|T) \ge R_0, \tag{44}$$

$$P_2^{\alpha}(T) \le \beta. \tag{45}$$

In the above, $P_2^{\alpha}(T)$ is defined by (23) and β is a constant representing allowed risk level by the software project's management.

The solution procedures to Formulations 4 and 5 can be established by joint analyses of the EC, $P_2^{\alpha}(T)$, and $R(x|T)$ functions, which are all functions of software release time T. For example, if the cost model used is the generalized software cost model (31), then the solution procedures for Formulation 5 using $P_2^{\alpha}(T)$, $\alpha > 0$, in (45) are given as follows:

Procedure 1.

i. Obtain T_{TBD}^{*} ($T_{TBD}^{*} \ge 0$) which is the solution to (43)-(44);
ii. If $P_2^{\alpha}(T_{TBD}^{*}) \le \beta$, then
$T^{*} = T_{TBD}^{*}$; procedure ends.
else
a) Obtain the smallest value of $T(T > T_{TBD}^{*})$ which makes $P_2^{\alpha}(T) \le \beta$ satisfied; denote this value by T_{P2}^{1};
b) $T^{*} = T_{P2}^{1}$; procedure ends.

If the cost model used is the generalized software cost model (31), then the solution procedures for Formulation 4 using $P_2^{\alpha}(T)$, $\alpha > 0$ in (45) are given as follows:

Procedure 2.

i. Obtain T_{TBD}^{*} ($T_{TBD}^{*} \ge 0$), which is the solution to (41);
ii. If P_2^{α} $(T_{TBD}^{*}) \le \beta$, then
$T^{*} = T_{TBD}^{*}$; procedure ends.
else
a) Obtain the smallest value of $T(T > T_{TBD}^{*})$, which makes $P_2^{\alpha}(T) \le \beta$ satisfied;
denote this value by T_{P2}^{1};
b) Obtain the largest value of T $(0 \le T < T_{TBD}^{*})$, which makes $P_2^{\alpha}(T) \le \beta$ satisfied; denote this value by T_{P2}^{2};
c) If $E[C(T_{P2}^{1})] < E[C(T_{P2}^{2})]$, then
$T^{*} = T_{P2}^{1}$; procedure ends.
else if $E[C(T_{P2}^{1})] > E[C(T_{P2}^{2})]$, then
$T^{*} = T_{P2}^{2}$; procedure ends.
else
$T^{*} = T_{P2}^{1}$ or $T^{*} = T_{P2}^{2}$, i.e., there are two solutions to the optimal software release problem; procedure ends.

The establishment of the above solution procedures is quite straightforward; thus, detailed explanations or proof of these procedures are omitted. The solution procedures

TABLE 3
Solutions to Example 5 under Different Requirements

Case No.	Requirements			Solution Obtained	
	α	β	R_0	T^*	$E[C(T^*)]$
#1	-	-	-	33	24854
#2	-	-	0.99	36	25106
#3	0.20	0.15	0.99	43	27155
#4	0.20	0.10	0.99	46	28396
#5	0.10	0.10	0.99	71	41104

under other situations for Formulations 4 and 5 (e.g., using the basic software cost model (24) or using $P_2^{\alpha}(T)$, $\alpha = 0$ in (45)) can be established in a similar manner.

Example 5. To illustrate, we reconsider Example 4, which takes Formulation 1 [11]. Now, we study this optimal software release problem under Formulation 5. The parameter values are given in Table 2 and (40).

Suppose the reliability requirement (44) is $R(x|T) \geq 0.99$ and, in the risk constraint (45), $\alpha = 0.20$ and $\beta = 0.10$. Using Procedure 1, the optimal software release time is found to be $T^* = 46$, under which $E[C(T^*)] = 28{,}396$, shown as Case #4 in Table 3.

It is interesting to compare the solution obtained with the one obtained in Example 4, which is shown as Case #1 in Table 3. In Example 4, there is no reliability requirement or risk constraint in the optimal software release problem, thus the EC can reach its minimum. If reliability requirement and/or risk constraint is imposed, normally the EC under the new solution will increase, with the reliability requirement and/or the risk constraint satisfied (Cases #2-#5 in Table 3). For example, if the reliability requirement of $R(x|T) \geq 0.99$ is added, then the optimal software release time becomes $T^* = 36$, under which $E[C(T^*)]$ increases by 1.0 percent, shown as Case #2 in Table 3. If a further risk constraint of $P_2^{0.2}(T) < 0.15$ is imposed, then the optimal software release time becomes $T^* = 43$, shown as Case #3 in Table 3, under which $E[C(T^*)]$ increases by another 8.2 percent.

However, it should be noted that management should not place too rigorous risk constraint on the optimal software release problem. For example, if the risk constraint is set to be $P_2^{0.1}(T) < 0.1$, then we have $T^* = 71$, under which $E[C(T^*)] = 28{,}396$, shown as Case #5 in Table 3. Compared with the solution obtained with the same reliability requirement but without any risk constraint (Case #2 in Table 3), the software release time and the EC increase by 97 percent and 64 percent, respectively, which may not be considered acceptable. Again, it is a matter of trade-off and whether this new solution obtained is viable is subject to management's judgment.

5.2 Maximization of Reliability with Cost Constraint

Besides the minimization of expected software cost (as Formulations 1 and 2), the optimal software release problem can also be formulated as a maximization of software reliability with a cost constraint (as Formulation 3). When uncertainty in software cost is considered, the cost constraint (5) that uses the EC should be modified into the one using the AC.

Formulation 6.

$$\text{Maximize } R(x|T), \tag{46}$$

$$\text{Subject to } P_3^{C_0}(T) \leq \beta, \tag{47}$$

where $P_3^{C_0}(T) \equiv \Pr\{C(T) > C_0\}$ is the risk that the AC of the software project is higher than the allowed budget, C_0. For the calculation of $P_3^{C_0}(T)$ under the basic software cost model (24), we have the following result:

Theorem 3. *If the software testing process is modeled by an NHPP SRM, then, under Assumptions 1 and 2, $P_3^{C_0}(T)$ for the basic software cost model (24) is*

$$P_3^{C_0}(T) = 1 - e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!}, \tag{48}$$

where

$$k_{up} \equiv \langle \gamma_3 \rangle, \quad j_k \equiv \left\langle \frac{c_1(\gamma_3 - k)}{c_2} \right\rangle, \quad \gamma_3 \equiv \frac{C_0 - c_3 T^{\kappa}}{c_1}. \tag{49}$$

Theorem 3 provides a close-form formulation to calculate $P_3^{C_0}(T)$ for the basic software cost model (24). The proof is similar to that of Theorem 2 and hence is omitted.

Example 6. Reconsider Example 2, which uses the basic software cost model (24) and for which the model parameters are shown in Table 1. The plots of $R(x|T)$ and $P_3^{C_0}(T)$ with different values of C_0 versus T are shown in Figs. 4a and 4b, respectively. We can observe the pattern of $P_3^{C_0}(T)$ curves; they appear to be roughly concave and approach 1 when T approaches infinity.

For other software cost models, due to model complexity, the analytical results for calculating $P_3^{C_0}(T)$ such as (48) may be difficult to derive. Similarly, Monte Carlo simulation can be used to obtain $P_3^{C_0}(T)$. For example, for the generalized software cost model, (31), for any given value of T, the simulation steps to calculate $P_3^{C_0}(T)$ are listed as follows:

1. Generate a value of n, which is a realization of a Poisson distributed random variable N_1 with mean $m(T)$. Generate a value of m, which is a realization of a Poisson distributed random variable N_2 with mean $m(T + T_w) - m(T_w)$. N_1 and N_2 are independent.
2. Generate $y_1, \cdots, y_n$, which are n independent realizations of a random variable which follows truncated exponential distribution with parameters λ_y and T_{0y}.
3. Generate $w_1, \cdots, w_m$, which are m independent realizations of a random variable, which follows truncated exponential distribution with parameters λ_w and T_{0w}.

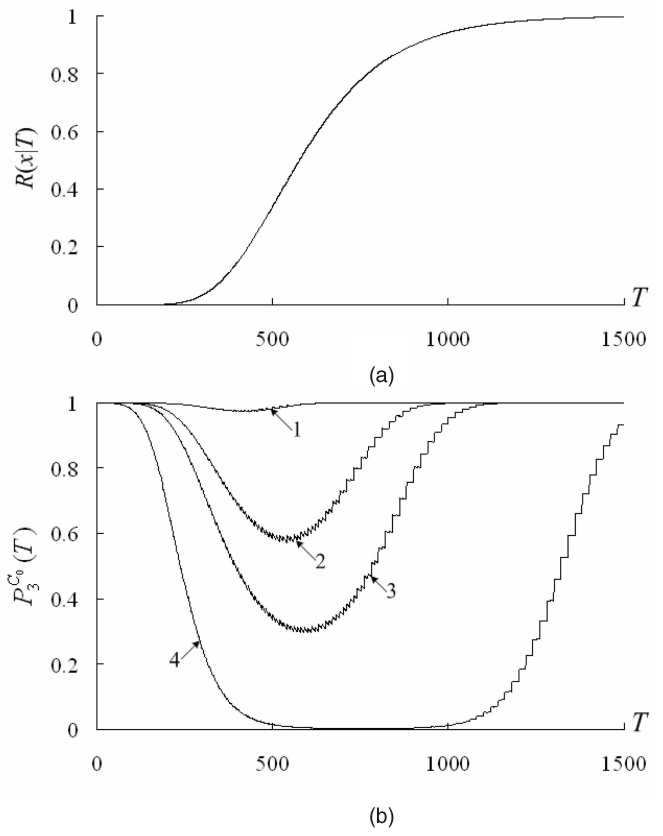

Fig. 4. Plots of (a) $R(x|T)$ and (b) $P_3^{C_0}(T)$ for the basic software cost model (24), using model parameters in Table 1. In (b), Curve 1: $C_0 = 10,000$, Curve 2: $C_0 = 13,500$, Curve 3: $C_0 = 15,000$, and Curve 4: $C_0 = 20,000$.

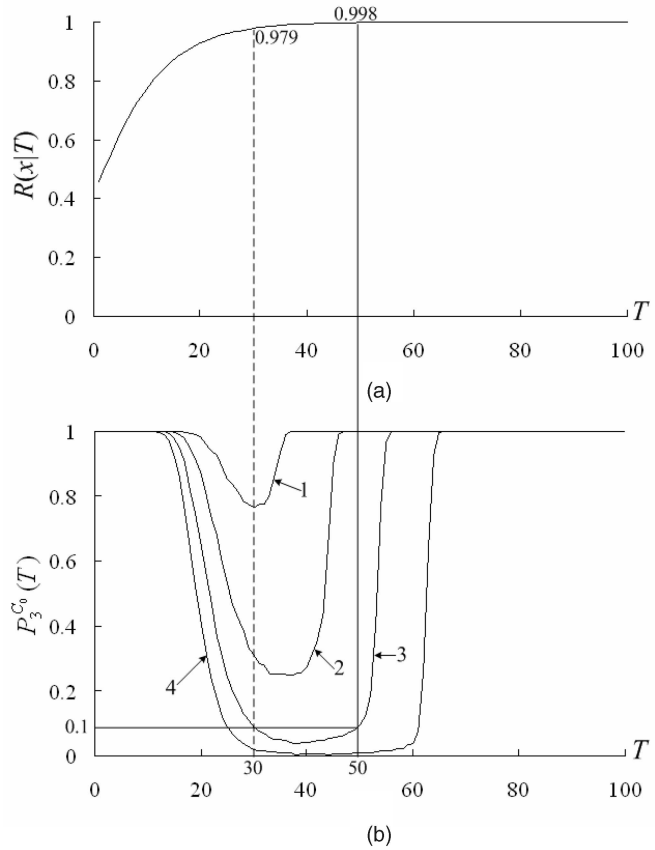

Fig. 5. Plots of (a) $R(x|T)$ and (b) $P_3^{C_0}(T)$ for the generalized software cost model (31), using model parameters in Table 2 and (40). In (b), Curve 1: $C_0 = 25,000$, Curve 2: $C_0 = 30,000$, Curve 3: $C_0 = 35,000$, and Curve 4: $C_0 = 40,000$.

4. Calculate

$$C(T) = c_0 + c_1 \sum_{i=1}^{n} y_i + c_2 \sum_{i=1}^{m} w_i + c_3 T^{\kappa} + c_4[1 - R(x|T)].$$

5. Repeat Steps 1-4 for l times, denote by l_0 the number of times that $C(T) > C_0$ happens, then l_0/l is an approximation of $P_3^{C_0}(T)$.

The calculation of $P_3^{C_0}(T)$ under other software cost models can be carried out in a similar fashion.

Example 7. Reconsider Example 4, which takes Formulation 1 and uses the generalized software cost model (31). Now, we study the optimal software release problem under Formulation 6. The parameter values are given in Table 2 and (40).

The plots of $E[C(T)]$ and $P_3^{C_0}(T)$ with different values of C_0 versus T are shown in Figs. 5a and 5b, respectively.

If in the cost constraint (47), $C_0 = 35{,}000$ and $\beta = 0.1$, then, from (47), it is obtained that $30 \le T^* \le 50$, as can be seen in Fig. 5b. Since the software reliability is a monotonous increasing function of T, the optimal software release time should be $T^* = 50$, under which $R(x|T^*) = 0.998$, shown as Case #5 in Table 4.

Table 4 presents solutions to the optimal software release problem in Example 6 under different values of C_0 and β.

From Examples 6 and 7, we can see that the solution procedures of the optimal software release problem taking Formulation 6 are quite straightforward, which are summarized as follows:

Procedure 3.

i. Obtain T_0 $(T_0 \ge 0)$ at which $P_3^{C_0}(T)(T \ge 0)$ reaches the minimum;
ii. If $P_3^{C_0}(T_0) > \beta$, then
    there is no solution to the optimization problem (46)-(47); procedure ends.
  else if $P_3^{C_0}(T_0) = \beta$, then
    $T^* = T_0$; procedure ends.
  else

TABLE 4
Solutions to Example 6 under Different Values of C_0 and β

Case No.	Requirements		Solution Obtained	
	C_0	β	T^*	$R(x \mid T^*)$
#1	25000	0.25	No solution	-
#2	30000	0.25	37	0.9912
#3	30000	0.20	No solution	-
#4	35000	0.20	52	0.9986
#5	35000	0.10	50	0.9983
#6	40000	0.10	61	0.9996

TABLE 5
Sensitivity Analysis Results for Example 5 When a and c_2 Change Separately

p	-30%	-20%	-10%	10%	20%	30%
$S_{p,a}$	-0.109	-0.087	-0.044	0.022	0.044	0.109
$Q_{p,a}$	-0.096	-0.074	-0.037	0.021	0.043	0.096
S_{p,c_2}	-0.065	-0.044	-0.022	0.022	0.023	0.022
Q_{p,c_2}	-0.055	-0.037	-0.018	0.018	0.020	0.023

a) Obtain the largest value of $T(T > T_0)$, which makes $P_3^{C_0}(T_0) \leq \beta$ satisfied; denote this value as T_1;
b) $T^* = T_1$; procedure ends.

5.3 Sensitivity Analysis

In our proposed formulations of the optimal software release problem, Formulations 4-6, there are SRM and software cost models involved which have a number of parameters that have to be estimated. The estimation can be done using collected data or by experiences from domain experts. In practice, estimation errors are inevitable, either due to insufficient data available or due to insufficient knowledge of experts. Therefore, it is important to conduct an analysis of the sensitivity of the result obtained with respect to the estimated parameters so that management can have some knowledge of the acceptability of the results obtained and attention can be paid to those parameters that affect the result significantly [33], [34], [35].

The sensitivity of the optimal software release time obtained, T^*, with respect to a model parameter, θ, can be quantified by $S_{p,\theta}$, which is the relative change of T^* when θ is changed by $100p$ percent [33], [34], i.e.,

$$S_{p,\theta} \equiv \frac{T^*(\theta + p\theta) - T^*(\theta)}{T^*(\theta)}. \tag{50}$$

Similarly, the sensitivity of expected software cost obtained, $E[C(T^*)]$, with respect to a model parameter, θ, can be quantified by $Q_{p,\theta}$, which is defined as

$$Q_{p,\theta} \equiv \frac{E\{C[T^*(\theta + p\theta)]\} - E\{C[T^*(\theta)]\}}{E\{C[T^*(\theta)]\}}. \tag{51}$$

We can also examine the effects of simultaneous changes of several model parameters [35]. For example, the sensitivity of T^* with respect to two model parameters, θ_1 and θ_2, can be quantified by $S_{p_1,p_2,\theta_1,\theta_2}$, which is the relative change of T^* when θ_1 is changed by $100p_1$ percent and θ_2 is changed by $100p_2$ percent, i.e.,

$$S_{p_1,p_2,\theta_1,\theta_2} \equiv \frac{T^*(\theta_1 + p_1\theta_1, \theta_2 + p_2\theta_2) - T^*(\theta_1, \theta_2)}{T^*(\theta_1, \theta_2)}. \tag{52}$$

Similarly, the sensitivity of $E[C(T^*)]$ with respect to two model parameters, θ_1 and θ_2, can be quantified by $Q_{p_1,p_2,\theta_1,\theta_2}$, which is defined as

TABLE 6
Sensitivity Analysis Results for Example 5 When a and c_2 Change Simultaneously and in the Same Direction

p_1	-30%	-20%	-10%	10%	20%	30%
p_2	-30%	-20%	-10%	10%	20%	30%
S_{p_1,p_2,a,c_2}	-0.196	-0.130	-0.065	0.022	0.087	0.130
Q_{p_1,p_2,a,c_2}	-0.165	-0.110	-0.055	0.024	0.079	0.117

TABLE 7
Sensitivity Analysis Results for Example 5 When a and c_2 Change Simultaneously and in the Reverse Direction

p_1	-30%	-20%	-10%	10%	20%	30%
p_2	30%	20%	10%	-10%	-20%	-30%
S_{p_1,p_2,a,c_2}	0.065	0.044	0.022	-0.022	-0.044	-0.065
Q_{p_1,p_2,a,c_2}	0.056	0.038	0.019	-0.019	-0.038	-0.058

$$Q_{p_1,p_2,\theta_1,\theta_2} \equiv \frac{E\{C[T^*(\theta_1 + p_1\theta_1, \theta_2 + p_2\theta_2)]\} - E\{C[T^*(\theta_1, \theta_2)]\}}{E\{C[T^*(\theta_1, \theta_2)]\}}. \tag{53}$$

The sensitivity of T^* and $E[C(T^*)]$ with respect to three or more model parameters can be studied in a similar way.

For illustrative purpose, we consider Example 5 discussed in Section 5.1. The optimal software release time obtained under the original values of model parameters is $T^* = 46$, under which $E[C(T^*)] = 28{,}396$. We conduct a sensitivity analysis for two model parameters, a (an SRM parameter) and c_2 (a cost model parameter). The results of the sensitivity analysis for other model parameters are not presented here as the purpose of this section is to highlight the importance of sensitivity analysis and to illustrate its procedures, rather than extensive sensitivity analysis.

Table 5 summarizes the results of sensitivity analysis for parameters a and c_2 when each parameter changes separately. It can be seen that the sensitivity of T^* and $E[C(T^*)]$ with respect to model parameters a and c_2 is at acceptably low levels, e.g., when a increases by 10 percent, the relative changes of T^* and $E[C(T^*)]$ are 2.2 percent and 2.1 percent, respectively; when c_2 decreases by 30 percent, the relative changes of T^* and $E[C(T^*)]$ are -6.5 percent and -5.5 percent, respectively. Results in Table 5 also reveal that a is a slightly more sensitive parameter than c_2.

Tables 6 and 7 summarize the sensitivity analysis results when parameters a and c_2 change simultaneously in the same direction and in the reverse direction, respectively. The results show acceptable sensitivity levels too. Therefore, it seems that T^* and $E[C(T^*)]$ obtained in Example 5 are trustworthy.

It can be noted that carrying out extensive sensitivity analysis for all model parameters requires much effort and is time consuming. Xie et al. [36] proposed conducting sensitivity analysis using the Design of Experiments (DOE) technique, which could be more efficient and could cater to interactions among model parameters. Interested readers can refer to [36] for more details.

6 Conclusion

Most of the existing research on the optimal software release problem gives insufficient consideration to the uncertainty (variability) of software cost and the formulizations of the problem are generally based on the treatment (such as minimization) of the EC. Since it is the AC, rather than the EC, that is meaningful to a specific software project, these formulations seem to have some flaws. If these formulations are used, then the solution obtained may give management a false dawn that the cost of this software project would be at a low level (e.g., has been minimized). In fact, what has been minimized or guaranteed to be below a certain level is the EC, not the AC; thus, there exists a certain level of risk that the AC of the project may be unexpectedly high and the project may run over budget.

In this paper, we study the uncertainty (variability) in software cost and its impact on optimal software release time in detail. It is clearly shown that there exists a certain level of uncertainty in the AC of a software project that should not be ignored. The uncertainty can be quantified either by the variance of the AC or by the risk functions defined in this paper. The risk functions for the basic software cost model (24) and for the generalized software cost model (31) are studied in detail by several examples. A risk-control approach to the optimal software release problem is proposed and new problem formulations are developed that are extensions of current formulations. Solution procedures for new formulations under certain conditions are established.

The main contribution of the research presented in this paper is to demonstrate the important fact that, in the optimal software release problem, the uncertainty involved in software cost should not be neglected. Based on this standpoint, we advocate adequate consideration of the uncertainty in software cost in future related research.

It is known that there exist many aspects of uncertainty in software cost/effort estimation [23], [24]. In this paper, we also propose conducting sensitivity analysis to study the uncertainty resulting from estimation of model parameters. Nevertheless, there are other aspects of uncertainty that have not been addressed in this paper, e.g., that result from unrealistic assumptions on which the SRM used is based. It can also be noted that, besides using risk functions to quantify the uncertainty in software cost, other methods, such as interval estimation, can be adopted as well. These could be our future research topics.

Appendix

Proof of Theorem 2

From (23), we have $P_2^{\alpha}(T) = 1 - f^{\alpha}(T)$, where

$$f^{\alpha}(T) = \Pr\{C(T) \le (1+\alpha)\cdot E[C(T)]\}. \tag{54}$$

Substituting (24) and (25) into (54), we have

$$\begin{aligned} f^{\alpha}(T) &= \Pr\left\{ \begin{array}{l} c_1 N(T) + c_2[N(\infty) - N(T)] + c_3 T^{\kappa} \le \\ (1+\alpha)\cdot\{c_1 m(T) + c_2[m_{\infty} - m(T)] + c_3 T^{\kappa}\} \end{array} \right\} \\ &= \Pr\left\{ N(T) + \frac{c_2}{c_1}[N(\infty) - N(T)] \le \gamma_2 \right\}, \end{aligned} \tag{55}$$

where γ_2 is given by (30).

Because NHPP has independent increments, $N(T)$ and $N(\infty) - N(T)$ are independent random variables and follow the Poisson distribution with parameters $m(T)$ and $m_{\infty} - m(T)$, respectively. Therefore, from probability theory, we have

$$\begin{aligned} f^{\alpha}(T) &= \sum_{k + c_2 j/c_1 \le \gamma_1} \Pr\{N(T) = k\} \Pr\{N(\infty) - N(T) = j\} \\ &= \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \Pr\{N(T) = k\} \Pr\{N(\infty) - N(T) = j\} \\ &= \sum_{k=0}^{k_{up}} \sum_{m=0}^{j_k} \frac{[m(T)]^k}{k!} e^{-m(T)} \frac{[m_{\infty} - m(T)]^j}{j!} e^{-[m_{\infty} - m(T)]} \\ &= e^{-m_{\infty}} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_{\infty} - m(T)]^j}{j!}, \end{aligned} \tag{56}$$

where k, j are nonnegative integers and k_{up} and j_k are given by (30). Finally, we have

$$\begin{aligned} P_2^{\alpha}(T) &= 1 - f^{\alpha}(T) \\ &= 1 - e^{-m_{\infty}} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_{\infty} - m(T)]^j}{j!}, \end{aligned} \tag{57}$$

by which Theorem 2 is proven. □

Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (863 Program) under Contract Number 2006AA04Z180 and the Sichuan Provincial Project of Scientific and Technical Supporting Programs, China, under Contract Number 07GG012-002.

References

[1] S. Yamada, T. Ichimori, and M. Nishiwaki, "Optimal Allocation Policies for Testing-Resource Based on a Software Reliability Growth Model," *Math. and Computer Modelling*, vol. 22, nos. 10-12, pp. 295-301, 1995.
[2] M. Xie, *Software Reliability Modelling*. World Scientific, 1991.
[3] K. Okumoto and A.L. Goel, "Optimum Release Time for Software Systems, Based on Reliability and Cost Criteria," *J. Systems and Software*, vol. 1, no. 4, pp. 315-318, 1980.
[4] H.S. Koch and P. Kubat, "Optimal Release Time for Computer Software," *IEEE Trans. Software Eng.*, vol. 9, no. 3, pp. 323-327, May 1983.
[5] S.M. Ross, "Software Reliability: The Stopping Rule Problem," *IEEE Trans. Software Eng.*, vol. 11, no. 12, pp. 1472-1476, Dec. 1985.
[6] S. Yamada and S. Osaki, "Cost-Reliability Optimal Release Policies for Software Systems," *IEEE Trans. Reliability*, vol. 34, no. 5, pp. 422-424, Dec. 1985.

[7] N.D. Singpurwalla, "Determining an Optimal Time Interval for Testing and Debugging Software," *IEEE Trans. Software Eng.*, vol. 17, no. 4, pp. 313-319, Apr. 1991.
[8] Y.W. Leung, "Optimum Software Release Time with a Given Cost Budget," *J. Systems and Software,* vol. 17, no. 3, pp. 233-242, 1992.
[9] H. Pham, "A Software Cost Model with Imperfect Debugging, Random Life Cycle and Penalty Cost," *Int'l J. Systems Science,* vol. 27, no. 5, pp. 455-463, 1996.
[10] 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. Computers,* vol. 46, no. 2, pp. 216-221, Feb. 1997, doi:10.1109/12.565602.
[11] H. Pham and X. Zhang, "A Software Cost Model with Warranty and Risk Costs," *IEEE Trans. Computers,* vol. 48, no. 1, pp. 71-75, Jan. 1999, doi:10.1109/12.743412.
[12] M. Kimura, T. Toyota, and S. Yamada, "Economic Analysis of Software Release Problems with Warranty Cost And Reliability Requirement," *Reliability Eng. and System Safety,* vol. 66, no. 1, pp. 49-55, 1999.
[13] M. Xie and B. Yang, "A Study of the Effect of Imperfect Debugging on Software Development Cost," *IEEE Trans. Software Eng.*, vol. 29, no. 5, pp. 471-473, May 2003, doi:10.1109/TSE.2003.1199075.
[14] N. Morali and R. Soyer, "Optimal Stopping in Software Testing," *Naval Research Logistics,* vol. 50, no. 1, pp. 88-104, 2003.
[15] H. Pham, "Software Reliability and Cost Models: Perspectives, Comparison, and Practice," *European J. Operational Research,* vol. 149, no. 3, pp. 475-489, 2003.
[16] P.J. Boland and H. Singh, "A Birth-Process Approach to Moranda's Geometric Software-Reliability Model," *IEEE Trans. Reliability,* vol. 52, no. 2, pp. 168-174, June 2003, doi:10.1109/TR.2003.813166.
[17] Y.C. Chang and W.L. Hung, "Software Release Policies on a Shot-Noise Process Model," *Applied Math. and Computation,* vol. 171, no. 2, pp. 746-759, 2005.
[18] C.Y. Huang, "Cost-Reliability-Optimal Release Policy for Software Reliability Models Incorporating Improvements in Testing Efficiency," *J. Systems and Software,* vol. 77, no. 2, pp. 139-155, 2005.
[19] C.Y. Huang and M.R. Lyu, "Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency," *IEEE Trans. Reliability,* vol. 54, no. 4, pp. 583-591, Dec. 2005, doi:10.1109/TR.2005.859230.
[20] C.T. Liu and Y.C. Chang, "A Reliability-Constrained Software Release Policy Using a Non-Gaussian Kalman Filter Model," *Probability in the Eng. and Informational Sciences,* vol. 21, no. 2, pp. 301-314, 2007.
[21] P.J. Bolanda and N.N. Chuiv, "Optimal Times for Software Release When Repair Is Imperfect," *Statistics and Probability Letters,* vol. 77, no. 12, pp. 1176-1184, 2007.
[22] B. Yang, H. Hu, and J. Zhou, "Optimal Software Release Time Determination with Risk Constraint," *Proc. 54th Ann. Reliability and Maintainability Symp.,* pp. 393-398, Jan. 2008.
[23] M. Jørgensen, "Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask," *IEEE Trans. Software Eng.*, vol. 30, no. 4, pp. 209-217, Apr. 2004, doi:10.1109/TSE.2004.1274041.
[24] M. Jørgensen, "Evidence-Based Guidelines for Assessment of Software Development Cost Uncertainty," *IEEE Trans Software Eng.*, vol. 31, no. 11, pp. 942-954, Nov. 2005, doi:10.1109/TSE.2005.128.
[25] W. Farr, "Software Reliability Software Survey," *Handbook of Software Reliability Engineering,* M.R. Lyu, ed., chapter 3, McGraw-Hill, 1996.
[26] J.D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application.* McGraw-Hill, 1987.
[27] J. Juran and F. Gryna, *Quality Control Handbook,* fourth ed. McGraw-Hill, 1988.
[28] S.A. Slaughter, E.D. Harter, and M.S. Krishnan, "Evaluating the Cost of Software Quality," *Comm. ACM,* vol. 41, no. 8, pp. 67-73, 1998.
[29] A.L. Goel and K. Okumoto, "Time Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Trans. Reliability,* vol. 28, no. 3, pp. 206-211, 1979.
[30] S. Yamada, M. Ohba, and S. Osaki, "S-Shaped Software Reliability Growth Models and Their Applications," *IEEE Trans. Reliability,* vol. 33, no. 4, pp. 289-292, Oct. 1984.
[31] M. Ohba, "Software Reliability Analysis Models," *IBM J. Research and Development,* vol. 28, no. 4, pp. 428-443, 1984.
[32] B. Yang and M. Xie, "A Study of Operational and Testing Reliability in Software Reliability Analysis," *Reliability Eng. and System Safety,* vol. 70, no. 3, pp. 323-329, 2000.
[33] M. Xie and G.Y. Hong, "A Study of the Sensitivity of Software Release Time," *J. Systems and Software,* vol. 44, no. 2, pp. 163-168, 1998.
[34] M. Xie and B. Yang, "Optimal Testing-Time Allocation for Modular Systems," *Int'l J. Quality and Reliability Management,* vol. 18, no. 8, pp. 854-863, 2001.
[35] C.Y. Huang and M.R. Lyu, "Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development," *IEEE Trans. Reliability,* vol. 54, no. 4, pp. 592-603, Dec. 2005, doi:10.1109/TR.2005.858099.
[36] M. Xie, B. Yang, and O. Gaudoin, "Sensitivity Analysis in Optimal Software Release Time Problems," *OPSEARCH,* vol. 41, no. 4, pp. 250-263, 2004.

Bo Yang received the BEng and the MEng degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1995 and 1998, respectively, and the PhD degree in software quality and reliability engineering from the National University of Singapore in 2002. He is currently a professor and the head of the Department of Industrial Engineering at the University of Electronic Science and Technology of China (UESTC), Chengdu, China. Before joining UESTC, he worked as a key member of the Software Engineering Process Group (SEPG) at Singapore Technology Aerospace and then worked as a postdoctoral research fellow in the Department of Computer Science at the National University of Singapore. His research interests include software reliability, software quality, distributed computing system, and Grid system. He has published 38 research papers. He is the coauthor of "Software Reliability Modeling and Analysis," a chapter in the *Encyclopedia of Statistics in Quality and Reliability* (John Wiley & Sons 2008). He is on the editorial board of the *Journal of Software Maintenance and Evolution: Research and Practice* and the *International Journal of Smart Home*. He serves as a referee for *IIE Transactions*, *Software Testing*, *Verification and Reliability*, *Computer Systems Science & Engineering*, *Asia Pacific Journal of Operational Research*, *Chinese Journal of Electronics*, *International Journal of Reliability*, *Quality and Safety Engineering*, etc. He has also served as a program committee member for 14 international conferences. He is a member of the IEEE and a senior member of the Chinese Institute of Electronics.

Huajun Hu received the BEng degree in information engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006. He is currently an MEng candidate in the Department of Information Engineering at UESTC. His research interests include software reliability, distributed computing system, and engineering optimization.

Lixin Jia received the BEng and the MEng degrees in electrical engineering from Xi'an Jiaotong University (XJTU), Xi'an, China, in 1989 and 1992, respectively, and the PhD degree in industrial intelligent control from XJTU in 2003. He is currently an associate professor in the School of Electrical Engineering at XJTU. His research interests include industrial intelligent control and engineering optimization. He has published more than 20 research papers. He won the Second Prize of the National Technical Invention Award of China in 2005.