Optimal Software Release Time Determination with Risk Constraint

Bo Yang, Ph.D., University of Electronic Science and Technology of China
Huajun Hu, University of Electronic Science and Technology of China
Jun Zhou, University of Electronic Science and Technology of China

Key Words: software reliability, release time, cost, risk constraint, NHPP models

SUMMARY & CONCLUSIONS

For a software development project, when to stop testing the software and release it for operation is of great importance as it impacts both the software reliability and the total cost of the project. Software release time determination, therefore, has attracted a lot of research in the past two decades and several new cost models have been developed in the literature recently. In most research on this topic, the approach taken is to minimize the expected total cost (ETC) of the software project, or further consider the software reliability requirement. However, because the actual total cost (ATC) of the software project is a random variable, minimization of the ETC does not guarantee that the ATC will be near this minimum. In fact, there exists certain risk that the ATC may exceed the ETC to an intolerable extent, which, despite its importance, has not been addressed in most related research. In this paper, we study the above mentioned risk problem for software release time determination and propose a new approach which could be helpful for management to control the risk of the project being over-budget. A numerical example is given to illustrate the solution procedures of the proposed approach.

1 INTRODUCTION

In general, a software development process consists of the following four phases: specification, design, coding and testing. The main goal of testing is to improve the quality and reliability of the software product. During the testing phase, the software is tested using designed test cases and software failures are observed. By analyzing each software failure, the embedded software faults which caused the software failure can be identified and removed. As a result, the reliability of the software tends to increase as the testing proceeds. Since a high reliability level is desirable for the software product and in some cases, e.g., for safety-critical software, a certain reliability level is required by the customer, the software will normally be tested for a rather long time before it can reach a satisfactory reliability level and be released for operation.

In practice, software reliability is not the only concern in a software project. Although software testing can improve the reliability of the software product, it may also have negative impacts from the cost-benefit point of view. For example, a delay of the release of the software product will normally cause a loss of market share and hence leads to a reduced economic benefit of the product; and a delay in the delivery of the software product may incur penalty costs specified by the contract. Therefore, when to stop testing and release the software to the customer is a crucial issue in a software development project.

The optimal software release problem has attracted much attention and research [1]-[8]. The basic approach is to develop a cost model which represents the expected total cost through the whole life-cycle of the software product, denoted by $E[C(T)]$, where T is the release time of the software and $C(T)$ is the total cost under the release time. Therefore the optimal software release time, T^*, is the time that minimizes $E[C(T)]$, i.e., the solution to the following problem [6][7]:

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

This is an unconstrained optimization problem which can be solved either analytically or by numerical methods.

On the other hand, there often exists a reliability requirement for the software product, thus the optimal software release problem is more often formulated as a constrained optimization problem [2][3][6][9]:

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

$$\text{Subject to} \quad R(x \mid T) \geq R_0, \tag{2}$$

where $R(x \mid T)$ is the reliability of the software product if it is released at time T, and R_0 is the required reliability level.

Another formulation of the optimal software release problem is to maximize software reliability with the constraint that the expected total cost does not exceed the cost budget [1], i.e.,

$$\text{Maximize} \quad R(x \mid T) \tag{3}$$

$$\text{Subject to} \quad E[C(T)] \leq C_0, \tag{4}$$

where C_0 is the allowed budget for the software project.

Yamada and Osaki [9] have derived a general form of solution to the optimal software release problem formulated in (1)-(2). If the expected total cost $E[C(T)]$ is a convex function and the reliability $R(x \mid T)$ is an increasing function of T, then the optimal software release time is

$$T^* = \max\{0, T_0, T_1\}, \tag{5}$$

where T_0 is the time that minimizes $E[C(T)]$ and T_1 is the smallest value that satisfies (2).

Most existing research on the optimal software release problem take the formulation of either (1)-(2) or (3)-(4), and effort has been put mainly on developing more realistic cost models or applying new software reliability models (SRMs). However, as can be seen from (1) and (4), the cost considered is the *expected total cost* (ETC), either as the objective function or as a constraint in the optimization problem. In reality, a software development project cannot be repeated many times, thus we argue that to minimize the ETC (as in (1)) or to ensure that the ETC does not exceed the allowed budget (as in (4)) does not make much sense. This is because for a software project, the *actual total cost* (ATC) incurred, $C(T)$, is a random variable, its one realization (for this specific software project) may be quite far from its expected value, the ETC. Therefore, minimization of the ETC does not guarantee that the ATC will be near this minimum. In other words, there exists a *risk* that the ATC may exceed the ETC to an intolerable extent. To address this issue, a new modeling approach to optimal software release problem needs to be developed.

In this paper, we study the above mentioned risk problem in detail, and propose a new modeling approach which could be helpful for management to control the risk of the software project being over-budget. As the purpose of this research is not to thoroughly study the optimal software release problem under various cost models but to address the problem of a lack of risk consideration in most existing research on this topic, we only use the problem formulation (1)-(2) as an example. We also use a simple cost model and a simple SRM in our study. However, it should be noted that the results obtained in our research can be generally applied to most optimal software release problems.

The remainder of this paper is organized as follows. In section 2 we illustrate the deficiency of current approaches to modeling of the optimal software release problem. In section 3 a new modeling approach with risk constraint is proposed and a numerical example is presented. In the last section some concluding remarks are given.

Notation

$N(t)$	Cumulative number of observed software failures in time interval (0, t] in software testing
$m(t)$	Mean value function, $m(t) = E[N(t)]$
m_∞	Defined as $m_\infty \equiv \lim_{t\to\infty} m(t)$
T_{LC}	Life-cycle length of the software product
T	Software release time
T^*_{TBD}	Software release time obtained without risk constraint
T^*	Optimal software release time
$W(T)$	Total testing-effort spent in $(0,T]$
$R(x \mid T)$	Software reliability function when it is released at time T
R_0	Required software reliability level at its release
C_0	Allowed budget for the software project
$C(T)$	Actual total cost (ATC) if the software is release at T
$E[C(T)]$	Expected total cost (ETC) if the software is released at time T, which is the expectation of $C(T)$
$C_1(T)$	Cost incurred by fault removal activities in the testing phase
$C_2(T)$	Cost incurred by fault removal activities in the operational phase
$C_3(T)$	General cost of testing
c_1	Cost of removing a fault in the testing phase
c_2	Cost of removing a fault in the operational phase
c_3	Cost per unit time of testing
c_w	Cost of testing per unit testing-effort expenditure
a	A parameter of G-O model which is the expected number of software failures to be eventually observed
b	A parameter of G-O model which is the failure occurrence rate per fault
$P_1(T)$	A risk function which is the probability that the ATC exceeds the ETC if the software is released at time T
$P_2(T)$	A risk function which is the probability that the ATC exceeds the ETC by α percent if the software is released at time T
α	Allowed margin for ATC being higher than ETC
β	Allowed risk limit

2 DEFICIENCY OF CURRENT APPROACHES

2.1 A Simple Cost Model Based on NHPP SRMs

Software reliability models (SRMs) are used in conjunction with cost models in the analysis of the optimal software release problem. Among many SRMs proposed, Nonhomogeneous Poisson Process (NHPP) models (see, e.g., [8]) have been thoroughly studied in the literature and widely used in practice. For this type of models, the software failure process is assumed to follow an NHPP. Denote by $N(t)$ the cumulative number of observed failures until time t, then $N(t)$ follows the Poisson distribution with parameter $m(t)$, i.e.,

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

where $m(t) \equiv E[N(t)]$ is the *mean value function*.

A simple yet widely used cost model which forms the basis of many later developed cost models is

$$C(T) = C_1(T) + C_2(T) + C_3(T), \quad (7)$$

where $C(T)$ is the ATC of the software project, $C_1(T)$ 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).

If the testing process follows an NHPP with mean value function $m(t)$, then at release time T, there will be $N(T)$ software faults removed and $N(T_{LC}) - N(T)$ software faults remaining in the software, which will have to be detected and removed in the operational phase. Here T_{LC} is the life-cycle

length of the software product. It should be noted that for any fixed value of T, 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. For NHPP SRMs, the ATC model (7) can be simplified into the following form

$$C(T) = c_1 N(T) + c_2[N(T_{LC}) - N(T)] + c_w W(T), \quad (8)$$

where 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 $c_2 > c_1$. $W(T)$ is the total testing-effort spent in $(0,T]$ and c_w is the cost of testing per unit testing-effort expenditure [6][7]. Note that $C(T)$ is a random variable, and its expectation, the ETC, is

$$E[C(T)] = c_1 m(T) + c_2[m(T_{LC}) - m(T)] + c_w W(T). \quad (9)$$

The variance of $C(T)$ is

$$Var[C(T)] = c_1^2 m(T) + c_2^2[m(T_{LC}) - m(T)]. \quad (10)$$

In some related research, the following assumptions are made:

(i) The life-cycle of the software product is infinity;

(ii) The limit of $m(t)$ when t approaches infinity exists, denoted by m_∞;

(iii) $W(T)$ is a linear function of T, i.e., $W(T) = kT$.

These assumptions are not very restrictive and could be valid under real-life situations. Assumption (i) is made by the fact that generally speaking a software product will be used much longer than the time spent in its testing phase, hence the life-cycle of the software product can be assumed to be infinity [8]. Assumption (ii) is satisfied by NHPP models with finite failures, e.g., the Goel-Okumoto (G-O) model [10] and the *S*-shaped model [8]. Assumption (iii) is made for the sake of simplicity.

Under assumptions (i)-(iii), the ATC model (8) becomes

$$C(T) = c_1 N(T) + c_2[N(\infty) - N(T)] + c_3 T, \quad (11)$$

where $c_3 = c_w k$ is the cost per unit time of testing. Accordingly, the ETC model (9) becomes

$$E[C(T)] = c_1 m(T) + c_2[m_\infty - m(T)] + c_3 T, \quad (12)$$

and the variance of the ATC (10) becomes

$$Var[C(T)] = c_2^2 m_\infty - (c_2^2 - c_1^2) m(T). \quad (13)$$

Because $c_2 > c_1$ and $m(T)$ is a monotonous increasing function with $m(0) = 0$, it can be obtained from (13) that $Var[C(T)]$ is a monotonous decreasing function with T and

$$\begin{aligned} Var[C(T)]_{\max} &= c_2^2 m_\infty, \quad T = 0; \\ Var[C(T)]_{\min} &= c_1^2 m_\infty, \quad T = \infty. \end{aligned} \quad (14)$$

Equation (12) is the basic ETC model which has been widely used in research on the optimal software release problem (see [8], Chapter 8). Recently some researchers have modified or extended the ETC model, (12), to reflect more realistic situations. For example, Pham [2] proposed an ETC model which assumed that the life cycle of the software was a random variable and took into account the penalty cost. Pham and Zhang [3] developed a generalized ETC model which considered warranty cost and software risk cost due to software failures. In their later work [5], the authors proposed an ETC model which incorporated testing coverage information.

It can be seen that in most related research, the cost model used is the ETC of the software project. However, it is known that the ATC of the software project is a random variable, thus it may not be close to its expectation (the ETC) in any one realization. For the cost model given by (11), the variance of the ATC is given by (13), whose minimal value is given by (14). Very often $c_1^2 m_\infty$ has a rather big value, thus the variance of ATC would not be small. Therefore, management could not have sufficient confidence that the ATC of the software project will near its ETC. To summarize, we argue that to consider only the ETC in the study of optimal software release problem seems inadequate, the ATC should also be taken into consideration and an approach to control the risk of the ATC being much higher than the ETC needs to be developed.

2.2 An Illustrative Example

Consider the optimal software release problem formulated by (1)-(2). Without loss of generality, we use the ETC model in (12) in this example. Studies under other ETC models can be easily carried out in a similar manner.

For the SRM, we use the G-O NHPP model [10], for which the mean value function is

$$m(t) = a[1 - \exp(-bt)], \quad (15)$$

where a and b are constants which can be estimated from observed testing data. The G-O model is one of the earliest NHPP SRMs developed and has far-reaching influences on later software reliability modeling. For the G-O model, we have

$$m_\infty = a. \quad (16)$$

In this case, by minimizing the ETC given by (12), we obtain T_0 as follows [8]

$$T_0 = \frac{1}{b} \ln \frac{ab(c_2 - c_1)}{c_3}, \quad (17)$$

and the time T_1 which makes (2) satisfied is [11]

$$T_1 = \frac{1}{b} \ln \frac{abx}{\ln(1/R_0)}. \quad (18)$$

The optimal software release time can be then obtained by (5).

Numerical Example 1

For this example, we use the data set taken from [10], which consists of 26 time-between-failure data obtained in the testing phase. Based on the data, the NHPP model parameters are estimated to be $\hat{a} = 33.99$ and $\hat{b} = 0.00579$. For cost parameters, we use the same values as those in [11], i.e., c_1=200, c_2=1500, c_3=10; and we assume the same software reliability requirement as in [11], i.e., $R(100 \mid T) \geq R_0 = 0.8$. The optimal software release time is then obtained as $T^* = 774$, under which the ETC is $E[C(T^*)] = 1.50 \times 10^4$.

Although $E[C(T^*)]$ is almost minimized, the ATC, $C(T)$, is a random variable whose variance can be obtained by (13). When $T = T^*$, we have

$$Var[C(T^*)] = 2.21 \times 10^6,$$

which is rather large. Therefore, it is highly possible that for this specific software project, the ATC incurred will exceed its

ETC, i.e., there exists no small risk.

In order to conduct a quantitative analysis on the risk, we define a function

$$P_1(T) \equiv \Pr\{C(T) > E[C(T)]\}, \quad (19)$$

which represents the risk that the ATC of a software project is greater than the ETC. Naturally, the smaller the value of $P_1(T)$, the better.

Theorem 1. If the testing process follows an NHPP with mean value function $m(t)$ and m_∞ exists, then for the ATC model given by (11), $P_1(T)$ is

$$P_1(T) = 1 - e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!}, \quad (20)$$

and its limit is

$$\lim_{T\to\infty} P_1(T) = 1 - e^{-m_\infty} \sum_{k=0}^{\langle m_\infty \rangle} \frac{(m_\infty)^k}{k!}. \quad (21)$$

In the above,

$$k_{up} = \langle \gamma_1 \rangle, \; j_k = \langle \frac{c_1(\gamma_1 - k)}{c_2} \rangle, \; \gamma_1 = \frac{c_1 m(T) + c_2[m_\infty - m(T)]}{c_1}, \quad (22)$$

and the notation $\langle x \rangle$ represents the largest integer value that is less or equal to x. The proof is given in the Appendix.

The $P_1(T)$ for Numerical Example 1 is shown in Figure 1. It can be seen that $P_1(T)$ does not change much when T changes, and the value of $P_1(T)$ falls in the interval (0.43, 0.53). From (21) we obtain the limit of $P_1(T)$ as $P_1(\infty) = 0.45$.

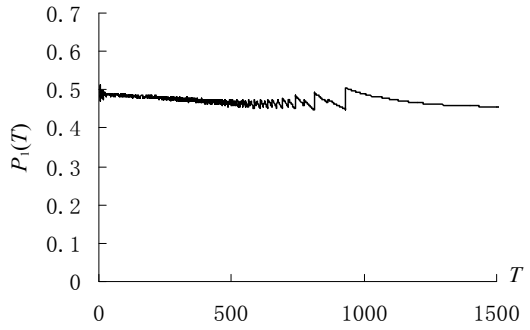

Figure 1: Plot of $P_1(T)$ for Numerical Example 1

The risk corresponding to $T^* = 774$ is $P_1(T^*) = 0.47$. This means if we take the optimal software release time as 774, obtained by solving the optimization problem of (1)-(2), then although the ETC of the software project is nearly minimized, which is 1.50×10^4, there is a probability of 0.47 that the ATC incurred will exceeds this value. In other words, management can have only a 53% confidence that the ATC will be less than or equal to 1.50×10^4, which is, obviously, not good enough.

Therefore, there is a great need to develop a new approach to modeling the optimal software release problem, which shall address the risk control issue. In the next section we will discuss this in detail.

3 A NEW APPROACH WITH RISK CONSTRAINT

Based on the above discussion and taking the risk control issue into consideration, we propose a new formulation of the optimal software release problem as follows.

$$\text{Minimize} \quad E[C(T)] \quad (23)$$

$$\text{Subject to} \quad R(x \mid T) \geq R_0, \quad (24)$$

$$P_2(T) < \beta, \quad (25)$$

where $P_2(T)$ is defined as

$$P_2(T) \equiv \Pr\{C(T) > (1+\alpha) \cdot E[C(T)]\}, \quad (26)$$

and β is the allowed risk level of the software project's management.

In (26), $P_2(T)$ is a risk function which is the probability (risk) that the ATC of the software project will exceed $(1+\alpha) \cdot$ ETC. Note that $P_1(T)$ can be viewed as a special case of $P_2(T)$ for which $\alpha = 0$. From our experiments we found (see Figure 1) that for the cost model (11), if we require the ATC not to exceed the ETC, the probability that this requirement is fulfilled is always not high enough. Therefore, management could relax this requirement a bit, i.e., allow the ATC to exceed the ETC to some extent, which is quantified by the coefficient $(1+\alpha)$.

It can be seen that the formulation given by (23)-(25) is actually a modification of (1)-(2), with another constraint added to the optimization problem. The added constraint reflects the requirement of risk control of the software project. Using the same idea, the original formulation of the optimal software release problem given by (3)-(4) can be modified into

$$\text{Maximize} \quad R(x \mid T) \quad (27)$$

$$\text{Subject to} \quad P_3(T) < \beta, \quad (28)$$

where $P_3(T)$ is defined as

$$P_3(T) \equiv \Pr\{C(T) > C_0\}. \quad (29)$$

It should be noted that the proposed formulations (23)-(25) and (27)-(28) are generic as they are applicable to any type of cost models as well as to any SRM. However, due to the limited length of this paper, we will not go into the details of various cost models and we only take the formulation (23)-(25) as an example to illustrate the ideas behind the methodology.

To solve the optimization problem given by (23)-(25), we need first to have some knowledge of the property of the risk function $P_2(T)$. Theorem 2 gives the results we obtain.

Theorem 2. If the testing process follows an NHPP with mean value function $m(t)$ and m_∞ exists, then for the ATC model given by (11), $P_2(T)$ is

$$P_2(T) = 1 - e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!}, \quad (30)$$

where

$$k_{up} = \langle \gamma_1 \rangle, \; j_k = \langle \frac{c_1(\gamma_1 - k)}{c_2} \rangle, \quad (31)$$

$$\gamma_1 = \frac{(1+\alpha)c_1 m(T) + (1+\alpha)c_2[m_\infty - m(T)] + \alpha c_3 T}{c_1}. \quad (32)$$

The proof is similar to that of Theorem 1 and thus is omitted.

Numerical Example 2

For the same parameter values as used in Numerical

Example 1, the plot of $P_2(T)$ with $\alpha = 0.10$ is shown in Figure 2(b). We can see that the risk, defined by (26), is not very high for any value of T. If we take the optimal software release time as 774, obtained from solving (23)-(24), then the risk of the ATC exceeding ETC by 10% is 0.147, obtained from (30). In other words, if the software is released at time 774, the software project management can have an 85.3% level of confidence that the ACT will not exceed the ETC by 10%. This situation is illustrated as the dashed lines in Figure 2.

If management is still not satisfied with the risk level (0.147), then he or she can assign a value of β in the optimization problem (23)-(25), e.g., set $\beta = 0.10$. In this case, the optimal software release time is found to be $T^* = 948$, illustrated as the real lines in Figure 2. The ETC of the software project under $T^* = 948$ is 1.65×10^4, i.e., an increase by 10%, which may be considered acceptable.

The above numerical example demonstrates clearly the procedures for solving the optimal software release problem (23)-(25). Generally speaking, we can first obtain a release time T^*_{TBD} by solving the optimization problem (23)-(24), which can be done easily using the result given in (5); then by incorporating the value of $P_2(T^*_{TBD})$, we can judge if the risk level at T^*_{TBD} is satisfactory. If it is, then $T^* = T^*_{TBD}$; otherwise we need to obtain T^* by solving (25).

In the determination of the optimal software release time, management of the software project has to make trade-offs between risk reduction and the increase in the ETC. For example, in numerical example 2, if management wants to reduce the risk from 0.15 to 0.10, then the software release time will have to be extended from 774 to 948, which results in a 10% increase in the ETC. This could be considered acceptable. However, if the increase in the ECT will be quite large when trying to reduce the risk, then management may still wish to keep the optimal software release time as T^*_{TBD}.

CONCLUSIONS

In this paper, we study the optimal software release problem from a risk control perspective, which is of practical importance for real-life software development projects. In most related research, the widely adopted approach is to minimize the expected total cost, which, based on our research, seems inadequate for addressing all issues involved, such as the risk of the project going over-budget. In this paper, we study the deficiency of current approaches to the optimal software release problem in detail, and two new formulations are proposed. A numerical example is given to illustrate the solution procedures for one of the proposed formulations. Using the proposed approach, management can have a certain level of confidence that the actual total cost of the software project will not exceed the expected total cost to an intolerable extent.

In this paper, we have only conducted a preliminary study on the optimal software release problem considering the risk control. More in-depth research on this new approach is needed. For example, a thorough study on the property of the risk function $P_2(T)$ is essential, and more detailed solution procedures for (23)-(25) and (27)-(28) need to be developed. In our future research, we plan to address these issues and conduct research on this topic for other cost models.

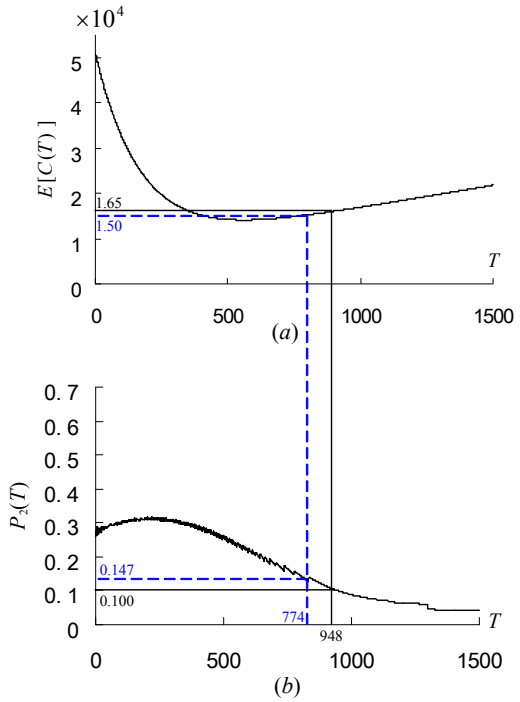

Figure 2: Software release time determination based on (23)-(25) using the same parameter values as in Numerical Example 1

Appendix Proof of Theorem 1

From (19) we have $P_1(T) = 1 - f(T)$, where

$$f(T) \equiv \Pr\{C(T) \le E[C(T)]\}.$$

Substitute (11) and (12) into $f(T)$, we have

$$f(T) = \Pr\left\{\begin{matrix} c_1 N(T) + c_2[N(\infty) - N(T)] + c_3 T \le \\ c_1 m(T) + c_2[m_\infty - m(T)] + c_3 T \end{matrix}\right\}$$

$$= \Pr\left\{N(T) + \frac{c_2}{c_1}[N(\infty) - N(T)] \le \gamma_1\right\}$$

where γ_1 is given by (22).

Because NHPP has independent increments, $N(T)$ and $N(\infty) - N(T)$ are independent random variables, and both $N(T)$ and $N(\infty) - N(T)$ follow Poisson distribution, with parameters $m(T)$ and $m_\infty - m(T)$, respectively. Therefore, using general theories on probability and NHPP, we have that

$$f(T) = \sum_{k + c_2 j / c_1 \le \gamma_1} \Pr\{N(T) = k\} \Pr\{N(\infty) - N(T) = j\}$$

$$= \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \Pr\{N(T) = k\} \Pr\{N(\infty) - N(T) = j\}$$

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

$$= e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!}, \qquad (33)$$

where k, j are nonnegative integers and k_{up}, j_k are given by (22). Therefore,

$$P_1(T) = 1 - f(T) = 1 - e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!} . \quad (34)$$

Now we derive the limit of $f(T)$. Note that

$$\lim_{T\to\infty} \gamma_1 = \lim_{T\to\infty} \frac{c_1 m(T) + c_2[m_\infty - m(T)]}{c_1} = m_\infty ,$$

thus $\lim_{T\to\infty} k_{up} = \langle m_\infty \rangle$, from (33) we have

$$\lim_{T\to\infty} f(T) = \lim_{T\to\infty} \left(e^{-m_\infty} \sum_{k=0}^{k_{up}} \sum_{j=0}^{j_k} \frac{[m(T)]^k}{k!} \frac{[m_\infty - m(T)]^j}{j!} \right)$$

$$= e^{-m_\infty} \lim_{T\to\infty} \sum_{k=0}^{k_{up}} \left(\frac{[m(T)]^k}{k!} \sum_{j=0}^{j_k} \frac{[m_\infty - m(T)]^j}{j!} \right)$$

$$= e^{-m_\infty} \lim_{T\to\infty} \left[\sum_{j=0}^{\frac{\langle c_1\gamma_1 \rangle}{c_2}} \frac{[m_\infty - m(T)]^j}{j!} + \sum_{k=1}^{k_{up}} \left(\frac{[m(T)]^k}{k!} \sum_{j=0}^{j_k} \frac{[m_\infty - m(T)]^j}{j!} \right) \right]$$

$$= e^{-m_\infty} \left[\begin{array}{l} \lim_{T\to\infty} \left(1 + \sum_{j=1}^{\frac{\langle c_1\gamma_1 \rangle}{c_2}} \frac{[m_\infty - m(T)]^j}{j!} \right) + \\ \lim_{T\to\infty} \sum_{k=1}^{k_{up}} \left(\frac{[m(T)]^k}{k!} \left(1 + \sum_{j=1}^{j_k} \frac{[m_\infty - m(T)]^j}{j!} \right) \right) \end{array} \right]$$

$$= e^{-m_\infty} \left(1 + \sum_{k=1}^{\langle m_\infty \rangle} \frac{(m_\infty)^k}{k!} \right) = e^{-m_\infty} \sum_{k=0}^{\langle m_\infty \rangle} \frac{(m_\infty)^k}{k!}$$

Finally, we have

$$\lim_{T\to\infty} P_1(T) = 1 - \lim_{T\to\infty} f(T) = 1 - e^{-m_\infty} \sum_{k=0}^{\langle m_\infty \rangle} \frac{(m_\infty)^k}{k!} . \quad (35)$$

By (34) and (35) Theorem 1 is proved.

REFERENCES

1. Y.W. Leung, "Optimum software release time with a given cost budget", *Journal of Systems and Software*, vol. 17, (Mar.) 1992, pp. 233-242.
2. H. Pham, "Software cost model with imperfect debugging, random life cycle and penalty cost", *International Journal of Systems Science*, vol. 27, (May.) 1996, pp. 455-463.
3. H. Pham, X.M. Zhang, "A software cost model with warranty and risk costs', *IEEE Trans. Computers*, vol. 48, (Jan.) 1999, pp. 71-75.
4. H. Pham, "Software reliability and cost models: perspectives, comparison, and practice", *European Journal of Operational Research*, vol. 149, (Sep.) 2003, pp. 475-489.
5. H. Pham, X.M. Zhang, "NHPP software reliability and cost models with testing coverage", *European Journal of Operational Research,* vol. 145, (Mar.) 2003, pp. 443-454.
6. C.Y. Huang, "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency", *Journal of Systems and Software,* vol. 77, (Aug.) 2005, pp. 139-155.
7. C.Y. Huang, M.R. Lyu, "Optimal release time for software systems considering cost, testing-effort, and test efficiency", *IEEE Trans. Reliability,* vol. 54, (Dec.) 2005, pp. 583-591.
8. M. Xie, *Software Reliability Modelling,* Singapore, World Scientific Publisher, 1991.
9. S. Yamada, S. Osaki,."Cost-reliability optimal release policies for software systems", *IEEE Trans. Reliability,* vol. R-34, (Dec.) 1985, pp. 422-424.
10. A.L. Goel, K. Okumoto, "Time dependent error-detection rate model for software reliability and other performance measures", *IEEE Trans. Reliability,* vol. R-28, 1979, pp. 206-211.
11. B. Yang, M. Xie, "A study of operational and testing reliability in software reliability analysis", *Reliability Engineering and System Safety*, vol. 70, (Dec.) 2000, pp. 323-329.

BIOGRAPHIES

Bo Yang (Corresponding Author)
School of Mechatronics Engineering
University of Electronic Science and Technology of China
Chengdu, Si Chuan, 610054, P.R.China

e-mail: yangbo@uestc.edu.cn

Dr. Bo Yang received B.Eng. and M.Eng. degrees in Electrical Engineering from Xi'an Jiaotong University in 1995 and 1998, respectively. He received a Ph.D. in Software Quality and Reliability Engineering from the Department of Industrial & Systems Engineering, National University of Singapore in 2002. Dr. Yang worked in Singapore Technology Aerospace Ltd. as a key member of the company's Software Engineering Process Group (SEPG) and later was a postdoctoral research fellow in the Software Engineering Lab., National University of Singapore. He joined the University of Electronic Science and Technology of China as an Associate Professor in 2003. His main research areas include software quality and reliability, system reliability analysis and optimization, distributed and Grid systems. He has published 33 research papers and coauthored a chapter for the book, Encyclopedia of Statistics in Quality and Reliability. He is on the Editorial Board of Int' Journal of Smart Home and referee for several international journals. He is a member of IEEE and a senior member of Chinese Institute of Electronics. He is currently the Head of Department of Industrial Engineering.

Huajun Hu, Jun Zhou
School of Electronic Engineering
University of Electronic Science and Technology of China
Chengdu, Si Chuan, 610054, P.R.China

Huajun Hu and Jun Zhou received B.Eng. degrees in Information Engineering from University of Electronic Science and Technology of China in 2006 and 2005, respectively. They are currently M.Eng. candidates the same program.