2011-9-15

Postrelease Testing and Software Release Policy for Enterprise-Level Systems

Zhengrui Jiang, *et al.*

Information Systems Research
Vol. 23, No. 3, Part 1 of 2, September 2012, pp. 635–657
ISSN 1047-7047 (print) | ISSN 1526-5536 (online)

http://dx.doi.org/10.1287/isre.1110.0379

Postrelease Testing and Software Release Policy for Enterprise-Level Systems

Zhengrui Jiang
College of Business, Iowa State University, Ames, Iowa 50011,
zjiang@iastate.edu

Sumit Sarkar, Varghese S. Jacob
School of Management, The University of Texas at Dallas, Richardson, Texas 75080
{sumit@utdallas.edu, vjacob@utdallas.edu}

Prior work on software release policy implicitly assumes that testing stops at the time of software release. In this research, we propose an alternative release policy for custom-built enterprise-level software projects that allows testing to continue for an additional period after the software product is released. Our analytical results show that the software release policy with postrelease testing has several important advantages over the policy without postrelease testing. First, the total expected cost is lower. Second, even though the optimal time to release the software is shortened, the reliability of the software is improved throughout its lifecycle. Third, although the expected number of undetected bugs is higher at the time of release, the expected number of software failures in the field is reduced. We also analyze the impact of market uncertainty on the release policy and find that all our prior findings remain valid. Finally, we examine a comprehensive scenario where in addition to uncertain market opportunity cost, testing resources allocated to the focal project can change before the end of testing. Interestingly, the software should be released earlier when testing resources are to be reduced after release.

Key words: software reliability; market opportunity cost; market uncertainty; learning; Bayes risk principle
History: Ram Gopal, Senior Editor; Giri Kumar Tayi, Associate Editor. This paper was received on February 5, 2009, and was with the authors 13 months for 4 revisions. Published online in *Articles in Advance* September 15, 2011.

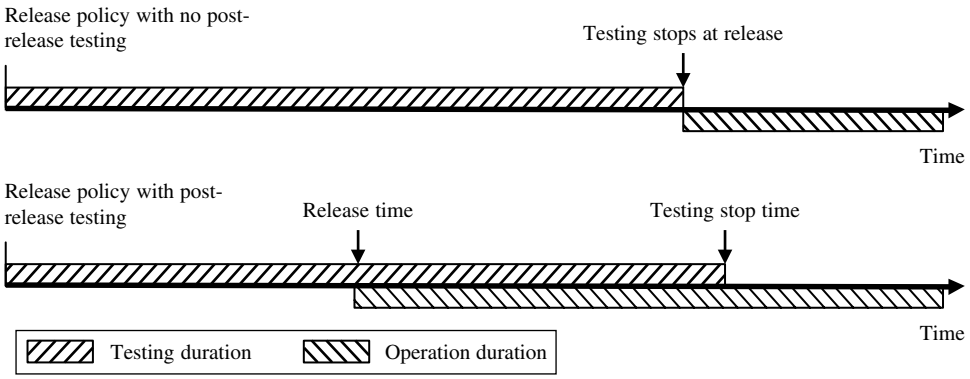
1. Introduction

Virtually all organizations rely on information systems to support their daily activities, improve the efficiency of business processes, or explore new business opportunities. Take the financial industry as an example. To reduce operating costs, most banks allow customers to manage their accounts, access monthly statements, and pay their bills online. To expand their customer base, some financial institutions now offer money market accounts or online stock brokerage services to customers who are difficult to reach with traditional products or services. Business operations of these types are unthinkable without the support of modern information systems. Unlike consumer or small business software, most complex enterprise-level information systems cannot be built by simply purchasing and installing a packaged mass-market product. When a firm's computing needs or existing infrastructure is unique, customized development may be the only viable option. Even if some components of the systems can be purchased, significant customization and integration efforts are typically still required. We refer to such systems as *custom-built enterprise-level* information systems, which are developed to support an organization's own operations and are not for sale on the market.

Despite the advancements in information technology, such systems are becoming increasingly costly to build and maintain, given the range and complexities of tasks they support. Today's enterprise-level software systems can contain up to millions of lines of code and cost millions of dollars and years to develop. In fact, many large organizations now spend a significant portion of their annual budget on the development and maintenance of such systems. Because of the importance, complexity, and cost, when a new system is developed, various process and project management decisions can impact not only the success of the project, but also the firm's overall position in the marketplace.

The focus of this research is on the release policy of custom-built enterprise-level information systems. Although such a system is not build for sale, when to put it into operation is still a very important, and

Figure 1 Release Policies with and Without Postrelease Testing



often difficult, decision to make. Our objective is to develop a release policy that can help speed up the release while reducing the risk and overall cost of the system. The problem we consider is a typical decision a project manager needs to make toward the end of the development lifecycle, namely, when to stop testing and release the system. The decision is difficult because of the enormous risks associated with a too-early or too-late release decision. If testing stops too early, many critical bugs may remain undiscovered. The firm thus risks dealing with dissatisfied users and incurring a high cost if the system breaks down during operation. For instance, software bugs have caused system crashes to high profile firms such as eBay and AT&T (Gross et al. 1999). It is estimated that buggy software costs the U.S. economy $60 billion each year (Thibodeau 2002).

Reliability considerations would dictate that testing should continue until all bugs are identified and removed, but prolonged testing leads to delays in software release, which is costly as well. Prior research has shown that there is a diminishing return to continued testing efforts (Dalal and Mallows 1988, Pham 2000). Besides the cost of testing, market opportunity cost can constitute an even bigger portion of the cost of late release. Although the systems we are interested in are not built for sale, such systems are used by organizations to improve the efficiency of existing operations or to tap new market opportunities. Therefore, a delay in release results in a delay in reaping the benefits of the new system, such as cost reduction, improving market share, or enhancing customer satisfaction. In a competitive environment, if competitors introduce a similar system earlier, a firm will risk falling into a disadvantageous competitive position. For these reasons, all else being equal, a software system delivered earlier is considered more valuable. This is the primary reason that managers sometimes choose to release a system even though they know it may still contain undetected bugs (e.g., Baskerville et al. 2001). Determining optimal testing stop time and release time, therefore, requires analysis of the trade-off between improved software reliability and costs of late release.

The economic consequences of an ad hoc release decision could be enormous, and many attempts have been made to formulate software release policies (e.g., Okumoto and Goel 1980, Dalal and Mallows 1988, Ehrlich et al. 1993, Singpurwalla and Wilson 1994, Pham and Zhang 1999, McDaid and Wilson 2001, Arora et al. 2006, Rinsaka and Dohi 2006, Chiu et al. 2009). This stream of research on software release policies is closely related to the broader software reliability literature, summaries of which have been provided by Pham (2000, 2006). In all these studies, the optimal release time is determined based on a cost-benefit analysis—that testing should continue until the expected gain from the improved reliability does not justify the cost of continued testing. An implicit assumption made in all these studies is that the testing stop time and the software release time are the same; i.e., formal testing stops completely at the time of release. After release, the task of identifying software bugs is shifted to the users, and bugs are identified and fixed only if they cause problems to users.

In this paper, we treat software release time and testing stop time as two separate decision variables. Within this context, we consider a new software release policy that explicitly allows for *active postrelease testing* (or *postrelease testing* for short).[1] As illustrated in Figure 1, the obvious difference between the release policy with no postrelease testing (the *NPT policy*) and the release policy with postrelease testing

[1] We define active postrelease testing as the testing activities after a software product is released that are geared toward the detection of bugs not yet encountered by users. This is distinct from the additional testing effort that may be needed to discover the causes of failures reported after release.

(the *PT policy*) is that with the latter, there will be a period during which testers continue to test the software while the software is in operation.

The key contribution of our research is the modeling of postrelease testing and the analyses of the release policy with postrelease testing considered. Specifically, our research seeks to address several questions. First, when should a software product be released, given the trade-offs among market opportunity cost, cost of testing, and cost of software failures in the field? Is it worthwhile to continue testing after release? We show that the existence of the market opportunity cost makes postrelease testing beneficial. Second, how does the release time for the PT policy compare to that for the NPT policy? Also, how do the testing stop times compare between the two policies? As one would expect, we find that the software is released earlier in the PT policy than in the NPT policy. Surprisingly, however, the testing stop time for the PT policy occurs after the stop time for the NPT policy. This is true even though a larger number of bugs can be detected during the same period of time in the PT policy because of the joint efforts of testers and users after release. Third, we examine whether users are exposed to a greater risk from the earlier release of the software advocated by the PT policy. It may appear that the earlier release (i.e., more remaining bugs in the software at the time of release) would result in more failures in the field over the lifetime of the product. Interestingly, we show that that is not the case; instead, the expected number of failures in the field is lower under the PT policy. Fourth, we analyze the impact of uncertain market opportunity cost on the solution to the PT policy and propose a method based on the Bayes risk principle to address such uncertainty. All our prior findings remain valid, and the PT policy continues to vastly outperform the NPT policy. Finally, we analyze scenarios where in addition to uncertain market opportunity cost, testing resources allocated to the focal project can change before the end of testing. Surprisingly, we find that when testing resources are to be reduced after release and the remaining testers' cost effectiveness decreases or remains the same, the software should be released earlier.

There exist a few studies from the software release policy literature that appear to be related to our work. However, as discussed below, there are fundamental differences in the research questions we address and those addressed by these studies. Arora et al. (2006) justify the "release early and fix later" practice for commercial off-the-shelf software. However, they do not model the detection of bugs by users and testers and the related consequences, which is one of the key aspects of our work. Our work also provides a methodology to determine the optimal release time and testing stop time based on characteristics of the project and various cost factors, which Arora et al. do not address. Furthermore, they do not explicitly model the various cost factors such as the cost of testing, cost of software failure in the field, and market opportunity cost; therefore, they do not address the research questions of this study. The work by Rinsaka and Dohi (2006) extends the prior release policy literature by proposing a model to simultaneously determine the optimal testing period and planned maintenance period. The planned maintenance period refers to the time between the testing stop time (release time) and the time that the project team is dissolved, during which the project team is kept in place to cope with software failures that may occur. Similar to most prior software release studies, Rinsaka and Dohi do not model postrelease testing, nor do they consider the impact of market opportunity on the software release policy. The studies by Dalal and Mallows (1988), Singpurwalla and Wilson (1994), McDaid and Wilson (2001), and Chiu et al. (2009), among others, propose policies to determine the optimal release time by taking into account the market opportunity costs. However, none of them recognizes that incorporating market opportunity cost can result in a fundamentally different and superior policy with postrelease testing. Thus, these studies also do not address any of our research questions. Table 1 summarizes the key differences between this study and these existing papers.

Table 1 Comparison Between This Study and Prior Research

	This study	Arora et al. (2006)	Rinsaka and Dohi (2006)	Studies that consider opportunity cost*
Modeling postrelease testing	Yes	No	No	No
Providing methods to determine the optimal release time and testing stop time as two separate decision variables	Yes	No	No	No
Considering opportunity cost	Yes	No	No	Yes
Considering the impact of uncertain market opportunity cost	Yes	No	No	No
Considering the impact of testing resource reallocation	Yes	No	No	No

* Source of data: Dalal and Mallows (1988, 1990), Singpurwalla (1991), Vienneau (1991), Krishnan (1994), Singpurwalla and Wilson (1994), McDaid and Wilson (2001), and Chiu et al. (2009).

2. Reliability Assumptions

The first two key software reliability assumptions are commonly seen in the software reliability literature (Goel 1985; Pham 2000, 2006); the third one is specific to the environment we model in this study.

Assumption 1. *The detection of each bug in a software system is independent of the detection of others, and the total bug-detection rate at any time is proportional to the number of undetected bugs at that time.*

The first part of this assumption is used in practically all software reliability models to ensure tractability. Assumption 1 implies that the lifetime of each bug follows an independent and identical exponential distribution. For instance, if we denote the *failure rate* of each bug under testers' testing by λ, the distribution of the lifetime of each bug is given by

$$f(t) = \lambda e^{-\lambda t}.$$

Thus, the probability that a bug will be detected by time t equals

$$F(t) = 1 - e^{-\lambda t}. \tag{1}$$

Assumption 2. *Once a bug is detected, it is fixed perfectly without causing any additional errors.*

This assumption is adopted for mathematical simplicity. Suppose the expected number of bugs is N just before the start of testing.[2] With perfect debugging, the number reduces to $(N-1)$ after the first bug is detected and removed, $(N-2)$ after the second bug is removed, and so on. In the absence of perfect debugging, a discovered bug may not be perfectly removed; occasionally removing a known bug may even introduce new bugs. Over time, however, the expected number of bugs tends to decrease with the amount of time spent in testing. Hence, we can assume that a bug is removed with probability P after it is detected. Goel and Okumota (1979) show that imperfect debugging is equivalent to perfect debugging with transformed parameters $\tilde{N} = N/P$ and $\tilde{\lambda} = P\lambda$. Ohba and Chou (1989) go on to demonstrate that models with imperfect and perfect debugging are isomorphic; i.e., when the models are being fitted to testing data, although the parameters obtained using different models are different, the cumulative number of detections at any given point in time is expected to be the same under the two models. This implies that even if debugging is not perfect, the perfect debugging model can still be used because the parameter estimation procedure can incorporate the imperfect debugging factor with revised parameter values. Therefore, Assumption 2 is not as restrictive as it may appear, and the findings of this research remain valid even under imperfect debugging.

Assumptions 1 and 2 are also the underlying assumptions of the classic Goel-Okumoto nonhomogenous Poisson process model (Goel and Okumoto 1979). Because of its parsimonious setup and sound performance (Ehrlich et al. 1993, Wood 1996), the Goel-Okumoto model is frequently adopted in the release policy literature (e.g., Pham and Zhang 1999, McDaid and Wilson 2001, Xie and Yang 2003). We denote the expected number of undetected bugs at the time of release τ by $u(\tau)$. Based on the Goel-Okumoto model, we have

$$u(\tau) = Ne^{-\lambda\tau}. \tag{2}$$

One way to interpret Equation (2) is that the expected number of undetected bugs at the time of release equals the product of the number of bugs at the beginning of testing and the probability that a bug is not detected by the time of release.

Because postrelease testing and users' bug-detection behavior are important aspects of this study, we make another assumption in this context.

Assumption 3. *After release, if a bug is first detected by users during operation, it causes only one software failure in the field; if a bug is first detected by testers during postrelease testing, it is patched before it can cause any software failure in the field.*

The first part of the assumption follows from the perfect debugging assumption. Once a failure occurs, the relevant portion of the enterprise system will be down until the bug is located and perfectly removed. The second part is reasonable so long as the time it takes to debug and install the patch is much shorter than the expected amount of time it takes for users to detect that specific bug. We expect this to be true in most real-world scenarios because testers typically follow well-defined testing procedures and can communicate more efficiently with developers, thus making it easier to locate and fix bugs reported by testers.[3]

Note that testers' and users' bug-detection efficiencies, reflected by the bug failure rates under testers' testing and users' usage, are not expected to be equal. First, the numbers of users and testers are typically different. Second, the intensities of the work may be different for testers and users. We denote the failure

[2] Consistent with the literature, bugs considered here refer to those that can cause failure during operation and are not easy to detect; they do not include minor problems such as typographic errors in system outputs or hardware/software compatibility problems that can be easily detected at the time of installation.

[3] In rare situations where this assumption is violated, we need to estimate the probability (P) that a bug detected by testers also causes a software failure before it is patched. The expected cost saving per bug detected during postrelease testing can then be revised to $(1-P)$ of the cost saving obtained based on Assumption 3. Our analyses in this research remain valid after this revision.

rate of a bug under testers' testing by λ and that during users' usage by $\lambda_u = r\lambda$, where r is the ratio of bug failure rate under users' usage to that under testers' testing. In practice, λ and λ_u can be estimated based on experience with prior projects that are comparable in size and scope.

3. The Cost Model

To determine the optimal testing stop time and release time, we consider those costs that have an impact on these two decision variables. We identify four such cost factors: *cost of testing*, *cost of fixing a bug detected during testing*, *cost of a software failure in the field*, and *market opportunity cost*.

The *cost of testing* refers to the cost of testers' activities such as test planning, test case generation, test execution, and analysis of testing results. As in prior literature (e.g., in Ehrlich et al. 1993, Pham and Zhang 1999), we assume that the cost of testing is a linear function of the amount of testing time.

The *cost of fixing a bug detected during testing* is the direct cost of bug removal in the development team's workplace and is assumed to be linear with the total number of bugs that testers detected. This assumption is also commonly seen in the literature (e.g., Okumoto and Goel 1980, Ehrlich et al. 1993). Furthermore, we assume that the same team (testers and developers) is kept during the entire testing process; hence this cost remains the same during prerelease and postrelease testing.

The *cost of software failure in the field* is incurred when a failure occurs during operation. This cost includes the direct cost associated with identifying and fixing the bug, the loss of revenue due to system down time, and other costs, such as liability cost and loss of goodwill resulting from users' dissatisfaction. The cost of a software failure in the field is usually orders of magnitude higher than the cost of fixing the same bug if it is detected during testing. The total cost of software failures in the field is assumed to be proportional to the number of defects not discovered by testers. This assumption has been widely adopted by prior studies (e.g., Dalal and Mallows 1988, Ehrlich et al. 1993, McDaid and Wilson 2001).

The *market opportunity cost*, denoted by $m(t)$, refers to market-related cost or opportunity cost due to late release. A similar opportunity cost has also been considered by Dalal and Mallows (1988), Singpurwalla and Wilson (1994), McDaid and Wilson (2001), and Chiu et al. (2009). Software systems are used to support existing operations and/or explore new market opportunities. If a system is used to support existing operations, because the bulk of the development cost has already been incurred before the testing phase, a delay in release simply reduces the time during which an organization can reap the benefits of the new system. In this case, the expected benefit per unit of time can be assumed to be a constant once the system is put into operation, and the resulting loss due to late release is a linear function of the amount of delay. If the new system is used to tap new market opportunities, a delay in its release increases the chance that the market may be exploited by competitors. The diffusion of a new service or product[4] generally follows an S-shaped curve (Bass 1969); i.e., the total number of adoptions increases at an increasing rate until nearly half of the potential customers are reached. Therefore, all else being equal, the first mover will capture a larger market share, and the difference in market share between the competing firms increases at an increasing rate with the amount of difference in release time. Furthermore, because of the learning curve and economies of scale, a larger market share may further lead to reduced cost per product or per unit of service. Consequently, in such a scenario the cost of late release is a strictly convex function of the release time. Taking into consideration the support of existing operations (linear market opportunity cost) and/or the exploitation of new market opportunities (strictly convex market opportunity cost), we have $m'(t) > 0$, $m''(t) \geq 0$. The functional form $m(t)$ can be obtained from experts familiar with the environment the system is built to operate in. If it is difficult to directly specify $m(t)$, the firm can first estimate costs for different release time points and use these estimates to approximate the cost curve.

We assume that the cost of installing patches is negligible in this study for the following reasons. If a complete reinstallation is not required, the process may cause little or no interruption to the operation of the system. Even if a new installation is required, it may be scheduled during off-business hours, low-traffic hours, etc. In addition, advances in telecommunication technologies have made remote patching feasible. In certain situations, software developers can directly access machines in different geographical locations to make corrections to existing installations. In other situations, software patches and instructions can be sent to remote locations and the installations completed by local information technology support teams.

Once we know the various costs, the optimal release policy is obtained by minimizing the sum of the four costs. In the next two sections, we first examine the NPT policy and then derive the optimal release time and optimal testing stop time for the PT

[4] Note that the service or product does not refer to the software itself; instead, this refers to services or products enabled by the software system.

Figure 2 Model Parameters

N—Expected number of bugs to be eventually detected
λ—Bug failure rate due to testers' testing, also referred to as testers' bug-detection effectiveness
λ_u—Bug failure rate during users' usage, also referred to as users' bug-detection effectiveness
r—Ratio of bug failure rate under users' usage to that under testers' testing
τ—Release time
T—Testing stop time
$u(\tau)$—Expected number of undetected bugs at the time of release
k—Cost of testing per unit time
a—Expected cost of one software failure in the field (including the cost of bug fixing)
b—Expected cost of fixing a bug detected during testing
c—Difference between the expected cost of a software failure in the field and the expected cost of fixing a bug detected during testing (i.e., $c = a - b$)

policy. The parameters shown in Figure 2 are used in the derivation of the various cost factors.

4. Total Costs for the Two Release Policies

4.1. Release Policy with No Postrelease Testing

When postrelease testing is not considered, testing stops at the time of release, implying $\tau = T$. We derive the various cost factors for the NPT policy.

(i) The expected cost of testing, denoted by $C^1_{\mathrm{NPT}}(\tau)$, is assumed to be linear with the amount of testing time; i.e.,

$$C^1_{\mathrm{NPT}}(\tau) = k\tau.$$

(ii) The expected cost of fixing bugs detected during testing, denoted by $C^2_{\mathrm{NPT}}(\tau)$, is proportional to the expected number of detected bugs at the time of release; i.e.,

$$C^2_{\mathrm{NPT}}(\tau) = b[N - u(\tau)] = bN(1 - e^{-\lambda\tau}).$$

(iii) The expected cost of software failures in the field, denoted by $C^3_{\mathrm{NPT}}(\tau)$, is linear with the expected number of undetected bugs at the time of release; i.e.,

$$C^3_{\mathrm{NPT}}(\tau) = au(\tau) = aNe^{-\lambda\tau} = bNe^{-\lambda\tau} + cNe^{-\lambda\tau}.$$

(iv) The market opportunity cost $m(\tau)$, as discussed in §3, is a function of the release time and satisfies the convexity condition (including both strict convexity and linearity).

By adding the four costs, we have the expected total cost for the NPT policy:

$$C_{\mathrm{NPT}}(\tau) = bN + k\tau + cNe^{-\lambda\tau} + m(\tau). \tag{3}$$

Note that bN in (3) is a constant. Therefore, the optimal release time depends on c, the difference between the cost of a software failure in the field and the cost of fixing a bug detected during testing, and not on the absolute values of the two individual costs.

$C_{\mathrm{NPT}}(\tau)$ is a strictly convex function of τ. The optimal release time without considering postrelease testing, denoted by τ^*_{NPT}, can be obtained by minimizing (3) with respect to τ. If the solution is interior, the optimal solution can be obtained based on the first-order condition; otherwise, $\tau^*_{\mathrm{NPT}} = 0$. The expected cost associated with the optimal release time is denoted by $C^*_{\mathrm{NPT}} \equiv C_{\mathrm{NPT}}(\tau^*_{\mathrm{NPT}})$.

4.2. Release Policy with Postrelease Testing

As illustrated in Figure 1, when postrelease testing is considered, a software system is released before testing stops. Therefore, the search for the optimal solution involves determining the values of two decision variables: the optimal release time τ and the optimal testing stop time T. As shown in Figure 3, the time horizon is divided into three periods under the PT policy. We first determine the probability that a bug is detected in each of the three periods.

(i) The probability that a bug is detected in the first period $[0, \tau]$, denoted by $F_1(\tau, T)$, is given by

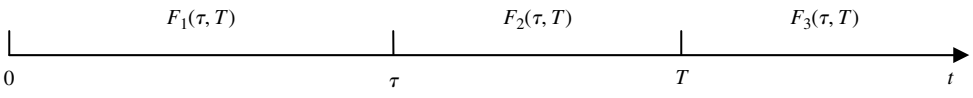
$$F_1(\tau, T) = 1 - e^{-\lambda\tau}.$$

(ii) In the second period $[\tau, T]$, both testers and users test the software; thus, the failure rate of a bug becomes $(r+1)\lambda$. The probability of a bug being detected in this period equals the probability that it is not found in the first period times the probability that the bug is detected in $(T - \tau)$ amount of time:

$$F_2(\tau, T) = e^{-\lambda\tau}(1 - e^{-(r+1)\lambda(T-\tau)}) = e^{-\lambda\tau} - e^{r\lambda\tau-(r+1)\lambda T}.$$

Note that in the second period, depending on who first detects a particular bug, the cost is different. The cost of a software failure in the field (a) is incurred if users first detect it, and a lower bug-fixing cost (b) is incurred if testers find it first. Based on Assumption 1

Figure 3 Different Probabilities of Bug Detection in Three Time Intervals



and the bug-detection ratio r, the lifetime of a bug under users' usage is an exponential random variable X_1 with failure rate $r\lambda$; that under tester's testing is an exponential random variable X_2 with failure rate λ. Given that a bug is detected in the second period, the probability that it is detected by the users before the testers is

$$P(\text{Detected by users} \mid \text{Detection in second period}) = \frac{r}{r+1}.^{5} \quad (4)$$

(iii) The probability that a bug is not detected until the last period equals

$$F_3(\tau, T) = e^{-\lambda\tau} e^{-(r+1)\lambda(T-\tau)} = e^{r\lambda\tau-(r+1)\lambda T}.$$

The expected number of bugs detected in each of the three periods equals the product of the total number of bugs and the three respective probabilities. The cost of testing, $C^1_{\mathrm{PT}}(\tau, T)$, the cost of fixing bugs detected during testing, $C^2_{\mathrm{PT}}(\tau, T)$, and the cost of software failures in the field, $C^3_{\mathrm{PT}}(\tau, T)$, are derived as:

$$\begin{cases} C^1_{\mathrm{PT}}(\tau, T) = kT, \\ C^2_{\mathrm{PT}}(\tau, T) = bN\left[F_1(\tau, T) + \dfrac{1}{r+1}F_2(\tau, T)\right], \quad \text{and} \\ C^3_{\mathrm{PT}}(\tau, T) = (b+c)N\left[\dfrac{r}{r+1}F_2(\tau, T) + F_3(\tau, T)\right]. \end{cases}$$

The market opportunity cost under the PT policy is a function of the release time τ. Incorporating the three derived probabilities into the above cost expressions and summing over all four types of costs, we obtain the total expected cost for the PT policy as

$$C_{\mathrm{PT}}(\tau, T) = bN + kT + \frac{cNr}{r+1}e^{-\lambda\tau} + \frac{cN}{r+1}e^{r\lambda\tau-(r+1)\lambda T} + m(\tau). \quad (5)$$

Note that $F_2(\tau, T)$—and hence $C_{\mathrm{PT}}(\tau, T)$—is not meaningful if $\tau > T$. However, we do not need to explicitly impose a constraint of $\tau \le T$ here because, as we show in the next section, this constraint is always satisfied by the optimal solution. Furthermore, we find that $C_{\mathrm{PT}}(\tau, T)$ has the following property.

Lemma 1. *The expected cost for the PT policy $C_{\mathrm{PT}}(\tau, T)$ is a strictly convex function of release time τ and testing stop time T.* (*Proofs of all lemmas and propositions are provided in the appendix.*)

[5] The derivation is shown in the appendix.

The strict convexity property of $C_{\mathrm{NPT}}(\tau)$ and $C_{\mathrm{PT}}(\tau, T)$ helps in analyzing the PT policy.

Under the PT policy, the optimal release time, denoted by τ^*_{PT}, and the optimal testing stop time, denoted by T^*_{PT}, can be simultaneously determined by minimizing $C_{\mathrm{PT}}(\tau, T)$. The associated minimum cost is $C^*_{\mathrm{PT}} \equiv C_{\mathrm{PT}}(\tau^*_{\mathrm{PT}}, T^*_{\mathrm{PT}})$. From (5), we again observe that the optimal solution depends on the value of c, the difference between the cost of a software failure in the field and the cost of removing a bug identified during testing, instead of the absolute values of these two individual parameters. In reality, the cost of a software failure in the field is typically orders of magnitude larger than the cost of fixing a bug detected during testing. Therefore, we have $c \approx a$. For expositional convenience, hereafter we refer to c as the *consequence* of a software failure in the field.

5. Evaluating the Polices with and Without Postrelease Testing

To begin with, we note that if $T = \tau$, $C_{\mathrm{PT}}(\tau, T)$ becomes $C_{\mathrm{NPT}}(\tau)$. This confirms that the NPT policy is a special case of the PT policy. We now derive the solutions for the two policies.

5.1. Optimal Solutions

We consider both interior and boundary solutions in this analysis. First, assume that all solutions are interior. For $C_{\mathrm{NPT}}(\tau)$, the first-order condition gives

$$k - \lambda cNe^{-\lambda\tau^*_{\mathrm{NPT}}} + m'(\tau^*_{\mathrm{NPT}}) = 0. \quad (6)$$

For $C_{\mathrm{PT}}(\tau,T)$, the two first-order conditions are

$$-\frac{\lambda cNr}{r+1}e^{-\lambda\tau^*_{\mathrm{PT}}} + \frac{\lambda cNr}{r+1}e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} + m'(\tau^*_{\mathrm{PT}}) = 0 \quad \text{and} \quad (7)$$

$$k - \lambda cNe^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} = 0. \quad (8)$$

Closed-form expressions cannot be obtained for τ^*_{NPT}, τ^*_{PT}, or T^*_{PT} in general, which makes it difficult to analyze the solutions in a straightforward manner. Therefore, additional algebraic transformations are needed to better understand the characteristics of the solutions to the two policies. Equation (8) can be rewritten as

$$\lambda cNe^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} = k. \quad (9)$$

Substituting for k into Equation (7), we have

$$-\frac{r}{r+1}\lambda cNe^{-\lambda\tau^*_{\mathrm{PT}}} + \frac{r}{r+1}k + m'(\tau^*_{\mathrm{PT}}) = 0. \quad (10)$$

Multiplying both sides of (10) by $(r+1)/r$, we obtain

$$k - \lambda cNe^{-\lambda\tau^*_{\mathrm{PT}}} + \frac{r+1}{r}m'(\tau^*_{\mathrm{PT}}) = 0. \quad (11)$$

Table 2 Relationship Between the Optimal Solutions for PT and NPT Policies

Intervals	I_1: $\lambda cN \le k$	I_2: $k < \lambda cN \le k + m'(0)$	I_3: $k + m'(0) < \lambda cN \le k + (r+1)/rm'(0)$	I_4: $\lambda cN > k + (r+1)/rm'(0)$
Solutions/Relationships	$\tau^*_{PT} = 0$	$\tau^*_{PT} = 0$	$\tau^*_{PT} = 0$	$T^*_{PT} > \tau^*_{NPT} > \tau^*_{PT} > 0$
	$\tau^*_{NPT} = 0$	$\tau^*_{NPT} = 0$	$T^*_{PT} > \tau^*_{NPT} > 0$	
	$T^*_{PT} = 0$	$T^*_{PT} = \frac{1}{(r+1)\lambda} \ln\left(\frac{\lambda cN}{k}\right) > 0$	$T^*_{PT} = \frac{1}{(r+1)\lambda} \ln\left(\frac{\lambda cN}{k}\right)$	

Note. Interior solutions not explicitly shown here are numerically obtained from Equations (6), (8), and (11).

Note that Equation (11) includes only τ^*_{PT} and is similar in form to Equation (6). Although it still cannot help us obtain a closed-form solution, Equation (11) makes it easier to compare the PT policy with the NPT policy and to help derive several interesting properties of the solutions.

It is also important to note that the optimal values obtained from the first-order conditions may not always be feasible; i.e., some of the values may be less than zero. In this case, the cost-minimizing feasible value is always zero, which is a boundary solution. After taking into consideration all solution scenarios, we arrive at the following conclusions:

Proposition 1. *If the market opportunity cost $m(\tau)$ equals zero, the optimal release time τ^*_{PT} and optimal testing stop time T^*_{PT} for the PT policy both equal τ^*_{NPT}, the optimal release time for the NPT policy; i.e., $\tau^*_{PT} = \tau^*_{NPT} = T^*_{PT}$.*

Proposition 1 shows that the market opportunity cost is what makes postrelease testing beneficial. In the absence of market opportunity cost, postrelease testing is not needed, and the optimal release time can be determined based on the NPT policy.

Proposition 2. *The optimal release time τ^*_{NPT} for the NPT policy and the optimal release time τ^*_{PT} and optimal testing stop time T^*_{PT} for the PT policy always satisfy $\tau^*_{PT} \le \tau^*_{NPT} \le T^*_{PT}$. When all solutions are interior, the strict inequality $\tau^*_{PT} < \tau^*_{NPT} < T^*_{PT}$ holds.*

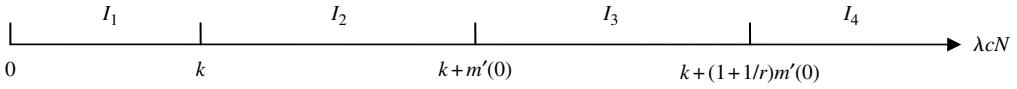
The finding that a software system should be released earlier under the PT policy than under the NPT policy is expected. With the PT policy, because postrelease testing helps reduce the risk faced by users, a firm can afford to release a system earlier to minimize market opportunity cost. The conclusion that testing should stop later under the PT policy than under the NPT policy may not be obvious at first. One may incorrectly think that when the PT policy is followed, because more bugs can be detected per unit of time as a result of the joint debugging efforts from testers and users, testing should stop earlier. The reason that testing should last even longer under the PT policy is as follows. As testing continues, the expected number of bugs detected in a unit of time and hence the expected benefit of testing decreases monotonically with time. In the meantime, the cost of testing and the market opportunity cost both increase with time.

Under the NPT policy, testing should stop when the marginal benefit of testing equals the sum of the marginal cost of testing and the marginal market opportunity cost. In contrast, when the PT policy is considered, testing should stop when the marginal benefit of testing equals just the marginal cost of testing. Because the cost as a result of continued testing after release is lower under the PT policy, it is optimal to test for a longer period of time.

The optimal values for τ^*_{NPT}, τ^*_{PT}, and T^*_{PT}, as well as their relationships, are summarized in Table 2. The detailed derivations are contained in the proof of Proposition 2 in the appendix.

Table 2 summarizes the relationship between the optimal solutions for the NPT and PT policies, and the conditions under which interior or boundary solutions apply. The intervals identified in the table reflect the comparisons between the marginal benefit and the marginal cost of delayed release or continued testing at time zero; the intervals are further illustrated in Figure 4. On the benefit side, λcN represents the *marginal benefit of testing*, becasue λN is the expected rate of bug detection by testers at time zero, and c is the cost saving for each bug detected by testers. On the cost side, k is the *marginal cost of testing* and $m'(0)$ is the marginal market opportunity cost at time zero. As shown in the first column of the table, when the marginal benefit cannot even cover the marginal cost of testing at time zero, i.e., $\lambda cN \le k$, the product should be released immediately without any testing, regardless of which policy is considered. In contrast, when the marginal benefit of testing is greater than the cost of testing, i.e., $\lambda cN > k$, then a positive amount of testing will be needed after release, if the PT policy is followed. Regarding the NPT policy, as shown in the first two columns of Table 2, the optimal release time should be zero if the marginal benefit of continued testing is less than or equal to the sum of the marginal cost of testing and marginal market opportunity cost at time zero, i.e., $\lambda cN \le k + m'(0)$. Otherwise, a positive amount of testing will be optimal under the NPT policy. Similarly, we can infer from the first three columns of the table that the optimal release time under the PT

Figure 4 Marginal Benefit from Testing (i.e., λcN) in Four Possible Intervals



policy should be zero if $\lambda cN \le k + [(r+1)/r]m'(0)$ and greater than zero otherwise. However, interpreting this condition is not as straightforward. By virtue of postrelease testing, only $r/(r+1)$ proportion of the bugs is expected to cause software failures in the field during the postrelease testing period. Furthermore, because of the involvement of users in bug detection, to achieve the same level of reliability, the cost of testing under the PT policy is only $r/(r+1)$ of that under the NPT policy. Therefore, in determining the optimal release time for the PT policy, we are actually trading off $[r/(r+1)]\lambda cN$ and $[r/(r+1)]k + m'(0)$, which is equivalent to comparing λcN with $k+[(r+1)/r]m'(0)$. This logic is also evident in the transformation from Equations (10) to (11).

We next show that the following *testing stop rule* is always valid.

Proposition 3. *Under the PT policy, it is optimal to stop testing once the expected number of undetected bugs drops to $k/(\lambda c)$.*

It is interesting to note that although the optimal release time and the optimal testing stop time depend on users' and testers' bug-detection rates and the various cost parameters, the expected number of undetected bugs at the optimal testing stop time is driven solely by the consequence of a software failure in the field (c) and the testers' cost effectiveness (represented by k/λ). Furthermore, we conclude from Proposition 3 that, in deciding whether to stop testing at a given time, all we need to know is the expected number of undetected bugs at that time: the testing history, i.e., when and by whom the known bugs were discovered, is not needed in the decision. Therefore, this testing stop rule provides an easily implementable way to decide whether testing should stop at a given time.

5.2. Comparing PT Policy with NPT Policy

Table 2 provides a quantitative comparison between the optimal solutions of the NPT and PT policies. In this section, we compare the solutions for the two policies along three dimensions that are of particular importance to a firm: (i) the expected cost, (ii) the reliability (represented by the expected number of undetected bugs) throughout the software lifecycle, and (iii) the expected number of software failures in the field. We ignore the case where $T^*_{PT} = \tau^*_{NPT} = \tau^*_{PT} = 0$ because the two policies are exactly the same for this extreme case, and furthermore, unlikely to occur in practice. Hereafter, we restrict our discussions to scenarios where the two policies do not lead to identical solutions.

Proposition 4. *The expected cost under the PT policy is strictly lower than that under the NPT policy; i.e., $C^*_{PT} < C^*_{NPT}$.*

This follows from the convexity of the cost functions, which implies the optimal solutions for both policies are unique. Further, because the NPT policy is a special case of the PT policy, and their optimal solutions (and hence the associated costs) are different, we must have $C^*_{PT} < C^*_{NPT}$.

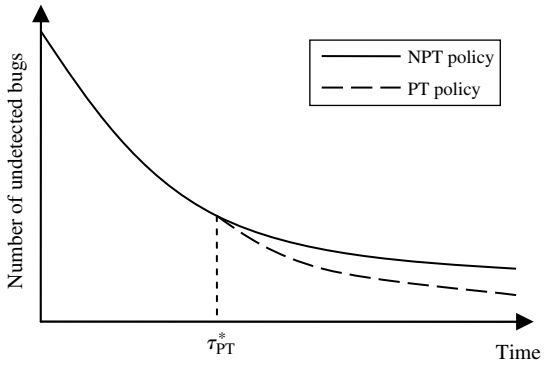
Proposition 5. *The expected reliability of the software obtained under the PT policy is strictly higher than that under the NPT policy from the release time of the PT policy.*

This is illustrated in Figure 5. Because under the PT policy users begin detecting bugs no later than they do under the NPT policy, and testers' testing lasts longer under the PT policy than under the NPT policy, the expected number of undetected bugs under the PT policy will never be higher than that under the NPT policy. In fact, the reliability curves start to separate from time τ^*_{PT} until the end of the lifecycle.

Proposition 6. *The expected number of software failures in the field under the PT policy is never larger than that under the NPT policy; when the market opportunity cost is strictly convex, the former is always lower than the latter.*

We believe that Proposition 6 is one of the most interesting findings of this research. Because the software is released earlier under the PT policy than under the NPT policy, one major concern decision makers may have regarding the PT policy is that users are exposed to a less-reliable software product and thus risk dealing with more software failures in the field during operation. Interestingly, Proposition 6 shows that the opposite is true. We find that this is an

Figure 5 Number of Undetected Bugs over Time



outcome of postrelease testing. Under the PT policy, although there are more undetected bugs when the product is released, all bugs will not result in software failures in the field to users—a portion of the bugs will be detected by testers during postrelease testing and be fixed before they cause problems to users. Furthermore, because testing lasts longer under the PT policy, the expected number of undetected bugs after testing stops is lower under this policy. These two factors jointly lead to the surprising result that there will be fewer expected software failures in the field under the PT policy than under the NPT policy.

6. Impact of Parameters

In this section, we examine how each individual model parameter impacts the optimal solution to the PT policy. We consider only interior solutions because they are more likely to occur in practice.[6] The following proposition summarizes the impact of the individual parameters on the solution of the PT policy.

Proposition 7. *When the optimal release time τ^*_{PT} and optimal testing stop time T^*_{PT} are interior,*

(i) *τ^*_{PT} and T^*_{PT} both increase with c—the consequence of a software failure in the field—and N—the number of undetected bugs at the start of testing: The duration of postrelease testing $(T^*_{PT} - \tau^*_{PT})$ increases with c and N when market opportunity cost is strictly convex and remains constant with c and N when market opportunity cost is linear.*

(ii) *τ^*_{PT} and T^*_{PT} both decrease with k—the cost of testing per unit time; $(T^*_{PT} - \tau^*_{PT})$ decreases with k when market opportunity cost is strictly convex and remains constant with k when market opportunity cost is linear.*

(iii) *τ^*_{PT} increases and $(T^*_{PT} - \tau^*_{PT})$ decreases with r—the ratio of bug failure rates under users' usage and testers' testing.*

(iv) *τ^*_{PT} and T^*_{PT} both decrease and $(T^*_{PT} - \tau^*_{PT})$ increases with $m'(\tau)$, the rate of increase in market opportunity cost.*

From Proposition 7, we conclude that mission-critical software (implying a larger c) should be tested more both before and after release. This is because with a higher cost of failure, more bugs need to be removed before exposing users to the risk. Further, based on the testing stop rule (Proposition 3), fewer bugs should remain after testing stops, thus requiring a longer overall testing duration. Similarly, when a software system is very complex or developed under time pressure (implying a larger N), more testing is needed before release to limit the risk to users, and it will take a longer testing time to reduce the number of undetected bugs to meet the testing stop condition described in Proposition 3.

[6] As one can see from Table 2, when a particular solution is at the boundary, a change in the value of a parameter within a certain range will not change the optimal solution.

The impact of the cost of testing (k) on the optimal solution is the opposite of the impact of c and N—with a higher testing cost, it is optimal to release early and to stop testing sooner. The shorter overall testing duration is expected. The earlier release time can be explained as follows. As discussed in §5.1, the optimal release time is the point where the marginal benefit and cost of delaying the release are equal. The marginal benefit is proportional to the instantaneous bug-detection rate and decreases monotonically with time. The marginal cost equals the sum of the cost of testing and the rate of increase in market opportunity cost. Because a higher value of k increases the marginal cost, the curves representing the marginal benefit and marginal cost intersect earlier, implying a quicker release.

The influence of r, the ratio between the bug failure rates under users' usage and testers' testing, is also not obvious. When there are more users or when the released software is used more frequently (implying a higher r), users are more likely to detect a bug before testers; to reduce the cost of failures in the field, testing should last longer before software release. Yet the optimal duration of postrelease testing shortens. Proposition 3 again helps explain the impact of r on the duration of postrelease testing. Based on this proposition, the expected number of undetected bugs at the optimal testing stop time does not change when r increases. Given that more bugs have been detected before release and users are more efficient at bug detection, it will take less time after release to reach the same level of reliability.

The impact of the rate of increase in market opportunity cost is the opposite to that of r. With a higher $m'(\tau)$, the software should be released earlier to reduce the market opportunity cost, and the software will be less reliable when first released. Again, based on Proposition 3, the expected number of undetected bugs at the optimal testing stop time does not change with $m'(\tau)$. Because there are more remaining bugs at the time of release, postrelease testing should last longer to reach the same level of reliability. Nevertheless, because users start contributing to bug detection earlier, the total duration of tester's testing is shortened because of the same reliability requirement at the end of testing.

7. Illustrative Example

In this section, we demonstrate how the models we propose can be used to determine the optimal release time and testing stop time. We adopt the set of parameters estimated based on testing data for a real-time control system (Pham 2006, pp. 144–145). The system contains approximately 200 modules; the average size of these modules is 1,000 lines of code. The number of initial bugs $N = 497$ and the failure rate $\lambda =$

0.0308 are estimated using the Goel-Okumoto model (Goel and Okumoto 1979). The cost parameters are set as follows: the cost of testing per unit of time $k = \$500$ per day; the cost of fixing a bug detected during testing $b = \$200$; and the cost of one software failure in the field $a = \$50{,}200$. The consequence of a software failure in the field c, i.e., the difference between the cost of a software failure in the field and the cost of fixing a bug detected during testing, equals \$50,000. The parameter r, the ratio of bug failure rates under testers' testing and users' usage, is first set equal to 1. We have tried other values, and the results are qualitatively similar. Regarding the market opportunity cost, we adopt a quadratic functional form, $m(\tau) = \theta \cdot (\tau + v)^2$, based on the opportunity cost proposed by Chiu et al. (2009). We first present results for $\theta = \$5{,}000$ and $v = 5.0$; results obtained using other values of θ and v are qualitatively consistent.

Substituting $m'(\tau) = 2\theta(\tau + v)$ into (6) and (11), we obtain the following solutions:

$$\tau^*_{\mathrm{NPT}} = -v - \frac{k}{2\theta} + \frac{1}{\lambda}\mathrm{Productlog}\left(\frac{c\lambda^2 N}{2\theta} e^{(k+2\theta v)\lambda/(2\theta)}\right),^{7} \tag{12}$$

$$\tau^*_{\mathrm{PT}} = -v - \frac{kr}{2\theta(1+r)} + \frac{1}{\lambda}\mathrm{Productlog}\left(\frac{c\lambda^2 Nr}{2\theta(1+r)} e^{((kr+2\theta v+2\theta vr)\lambda)/(2\theta(1+r))}\right). \tag{13}$$

Once τ^*_{PT} is determined, T^*_{PT} can be obtained based on (8):

$$T^*_{\mathrm{PT}} = \frac{1}{(r+1)\lambda} \ln \frac{\lambda c N}{k} + \frac{r}{r+1}\tau^*_{\mathrm{PT}}. \tag{14}$$

When a value obtained from these formulas is less than zero, the boundary solution zero is used instead.

Based on the given parameters, we obtain the following solutions: $\tau^*_{\mathrm{NPT}} = 27.64$ days, $\tau^*_{\mathrm{PT}} = 17.39$ days, and $T^*_{\mathrm{PT}} = 127.75$ days. The associated costs are $C^*_{\mathrm{NPT}} =$ \$16.05 million and $C^*_{\mathrm{PT}} =$ \$9.95 million. Under the PT policy, the system is released approximately 10 days earlier, and testing is extended by 100 days. By switching from NPT policy to PT policy, the firm achieves a cost saving of 38%.

This example can also be used to illustrate the result in Proposition 6. Based on the derivations in §4.2, we find that the expected number of undetected bugs at the time of release is 212 for the NPT policy and 291 for the PT policy. Under the NPT policy, all 212 undetected bugs will cause software failures in the field. But under the PT policy, the testers are expected to detect another 145 bugs during postrelease testing; hence the expected number of failures in the field reduces to 146. Therefore, the users actually face less risk under the PT policy than under the NPT policy, although the software is less reliable when first released under the PT policy.

The influences of the three important parameters c, θ, and r on the expected costs of the two policies are shown in Figures 6, 7, and 8, respectively. As we can see from Figures 6 and 7, the influence of c, the consequence of a software failure in the field, and θ, reflecting the magnitude of the market opportunity cost, are similar. In both cases, as the value of the respective parameter increases, the expected cost increases for both policies, and their difference widens. The impact of r, the bug-detection ratio between users and testers, is different. As we can see from Figure 8, the value of r has no impact on the expected cost of the NPT policy; this is because postrelease testing is not considered, so r is irrelevant. But the expected cost associated with the PT policy increases monotonically with r; hence the performance gap between the two policies narrows as r increases. In all three figures, the expected cost for the PT policy is lower than that for the NPT policy, which is consistent with Proposition 4.

Figure 6 Impact of the Consequence of a Software Failure in the Field

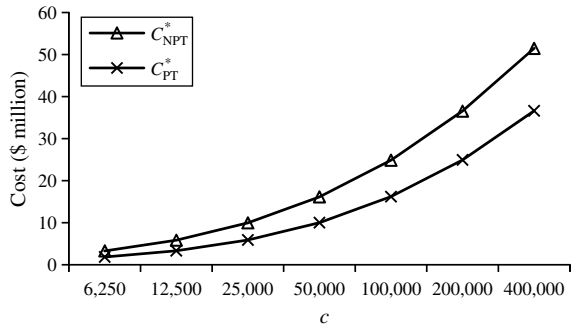
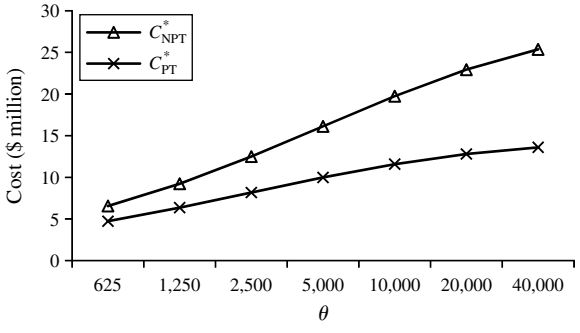
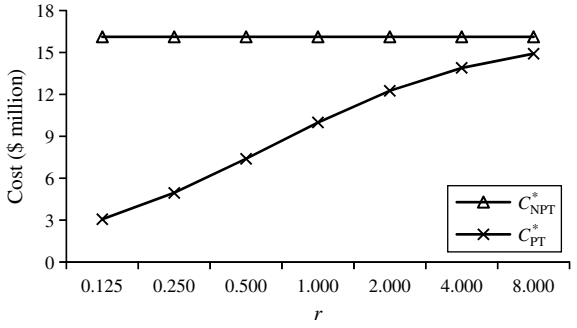


Figure 7 Impact of Market Opportunity Cost



[7] The *Productlog* function used here is the inverse function of $f(w) = we^w$. This function is not one of the *elementary functions* of calculus and has some complex mathematical properties. In this study, the argument of the function is always positive. Within this range, *Productlog* is a monotonically increasing function of its argument and has a unique real value.

Figure 8 Impact of Bug Failure Rate Ratio



During postrelease testing, although users and testers do not directly communicate with each other, their bug-detection activities interact because they target the same set of undetected bugs—if a bug is first detected by one group, the other group will not detect it. To better understand the impact of this interaction on the solution of the PT policy, we first vary the bug failure rate under users' usage (λ_u) while keeping that under testers' testing (λ) fixed, and then vary the bug failure rate caused by testers' testing (λ) while keeping that under users' usage (λ_u) fixed. The impact of such changes on the optimal release time, optimal testing stop time, and the duration of postrelease testing, are shown in Figures 9 and 10. From Figure 9, we conclude that as the users' bug-detection effectiveness (represented by λ_u) increases, the time to release (τ^*_{PT}) increases to reduce the risk of software failures in the field to users; the total testing time (T^*_{PT}), in contrast, decreases because the expected number of undetected bugs under the optimal solution can be reached earlier. Hence, the duration of postrelease testing ($T^*_{PT} - \tau^*_{PT}$) decreases. The impact of testers' bug-detection effectiveness (represented by λ) on the optimal solution, as shown in Figure 10, is not monotonic. When λ is small, τ^*_{PT}, T^*_{PT}, and $(T^*_{PT} - \tau^*_{PT})$ all tend to increase with λ. When λ is large, τ^*_{PT}, T^*_{PT}, and $(T^*_{PT} - \tau^*_{PT})$ all tend to decrease with λ.

The results can be explained by considering the two extremes of λ. At an extremely low value of λ, testing provides practically no benefits. Therefore, the firm should release the product immediately to minimize the market opportunity cost. As λ increases from an extremely low value, the gains from testing increase. We can infer from the testing stop condition that fewer bugs should remain at the end of testing, and therefore it becomes optimal to conduct additional testing to reduce the cost of software failures in the field. Yet when λ is extremely large, testers will be able to detect almost all bugs in a very short period of time. Therefore, prolonged testing again becomes unnecessary.

Figure 9 Impact of Users' Bug-Detection Effectiveness

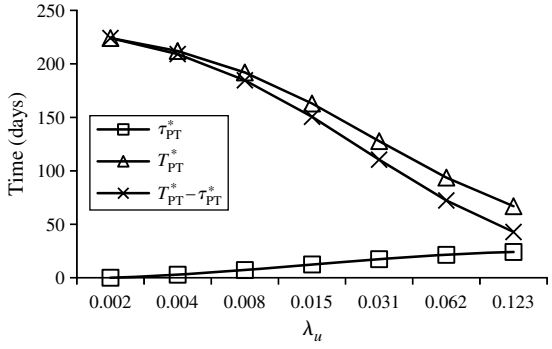
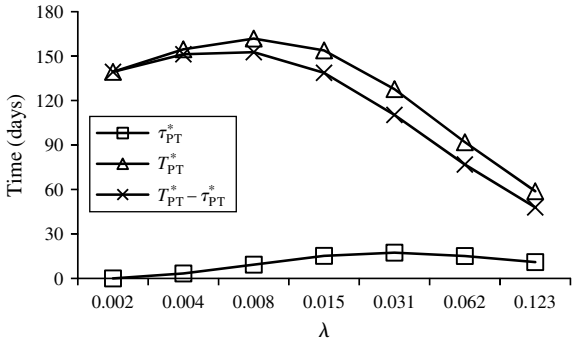


Figure 10 Impact of Testers' Bug-Detection Effectiveness



8. Uncertainty in Market Opportunity Cost

Because of the various uncertainties associated with the market size and competitors' market behavior, market opportunity is often the most difficult cost factor to estimate. For instance, if a competitor enters the market earlier than predicted, the market opportunity cost is likely to increase faster than originally projected. Similarly, if the market size is underestimated, the estimates for the market opportunity cost as well as its rate of increase will be lower than their true values. In this section, we examine the implications of uncertain market opportunity cost and develop methods to address such uncertainties.

8.1. Consequences of Over- or Underestimating Market Opportunity Cost

From Equation (11), we infer that the rate of increase in market opportunity cost, rather than the absolute value of the cost, determines the optimal release time. Further, we conclude based on Proposition 7 that if the estimated rate of increase in market opportunity cost is higher (lower) than the actual rate of increase, the recommended release time and testing stop time will both be earlier (later) than the optimal times, and the recommended duration of postrelease testing will be longer (shorter) than the optimal duration.

To illustrate the impact of uncertain market opportunity cost on the solution to the PT policy, we again adopt the quadratic market opportunity cost function $m(\tau) = \theta(\tau + v)^2$. We call θ the *scale* parameter, because $m(\tau)$ is proportional to θ at any point in time. The value of the scale parameter reflects, for instance, the potential market size for a new product or service. We name ν the *urgency* parameter because all else being equal, a higher ν value is equivalent to shifting a monotonically increasing cost curve to the left. The value of the urgency parameter can be affected, for instance, by a competitor's market entry time. The rate of increase in market opportunity cost, $m'(\tau)$, depends on both the scale parameter and the urgency parameter. All else being equal, $m'(\tau)$ increases with a higher θ or a higher ν.

Suppose that the true market opportunity cost function is $m(\tau) = 4{,}000 \cdot (\tau + 3.0)^2$. Using the same parameter values as in §7, we obtain the optimal release time and testing stop time as $\tau^*_{\mathrm{PT}} = 21.59$ days and $T^*_{\mathrm{PT}} = 129.85$ days; the optimal duration of postrelease testing is 108.26 days. If the market opportunity cost is projected to be $\hat{m}(\tau) = 6{,}000 \cdot (\tau + 5.0)^2$ (hence $\hat{m}'(\tau) > m'(\tau)$ for all τ), the recommended release time and testing stop time would be $\hat{\tau}^*_{\mathrm{PT}} = 15.05$ days and $\hat{T}^*_{\mathrm{PT}} = 126.59$ days, both earlier than the optimal times obtained based on the true market opportunity cost; the recommended duration of postrelease testing would be 111.54 days, which is longer than the optimal duration. Similarly, it can be shown that the opposite is true when the projected rate of increase in market opportunity cost is lower than the true rate of increase, i.e., $\hat{m}'(\tau) < m'(\tau)$.

8.2. Bayes Risk Approach to Determine Release and Testing Stop Times

Better information gathering and projection techniques seem to be the logical solutions to resolve uncertainty concerning the market opportunity cost, but obtaining a precise estimate of this cost could be difficult in practice. Therefore, we consider a method to reduce the risks resulting from the uncertain market opportunity cost using the *Bayes risk principle* (Berger 1993, p. 17). This approach leads to solutions that minimize the expected loss.

In our context, applying the Bayes risk principle results in the release and testing stop times that minimize the *expected* total cost. For expositional convenience, we consider the market opportunity cost a *random function*, denoted by $M(t)$, and use $m_i(t)$ and p_i to represent, respectively, the market opportunity cost under market scenario i and the probability associated with that scenario. The expected total cost, with the uncertain market opportunity cost considered, then equals:

$$E_{M(t)}[C_{\mathrm{PT}}(\tau, T)] = bN + kT + \frac{cNr}{r+1}e^{-\lambda\tau} + \frac{cN}{r+1}e^{r\lambda\tau-(r+1)\lambda T} + \sum_i p_i m_i(\tau).$$

If we denote the optimal release time and the optimal testing stop time obtained using the Bayes risk principle by τ^B_{PT} and T^B_{PT}, respectively, the following inequality must hold:

$$E_{M(t)}[C_{\mathrm{PT}}(\tau^B_{\mathrm{PT}}, T^B_{\mathrm{PT}})] \le E_{M(t)}[C_{\mathrm{PT}}(\tau, T)], \quad \forall\, 0 \le \tau < T.$$

Given the types of market opportunity costs we consider in this research (defined in §3), each $m_i(\tau)$ is still a convex and increasing function for each market scenario. Because it is the weighted sum of multiple convex and increasing functions, the expected market opportunity cost, $\sum_i p_i m_i(\tau)$, also remains a convex and increasing function of the release time, which leads to the following conclusion:

Observation 1. *When using the Bayes risk principle to address uncertain market opportunity cost, all the mathematical properties and findings regarding the PT policy remain valid.*

Therefore, by replacing the market opportunity cost $m(t)$ in (5) with the expected market opportunity cost, the optimal solution can still be computed using Table 2 and Equations (8) and (11). We call the solution thus obtained the *Bayes solution* to the PT policy.

Again based on the opportunity cost function proposed by Chiu et al. (2009), we let $m_i(\tau) = \theta_i(\tau + v_i)^2$. The expected market opportunity cost equals

$$E[M(\tau)] \equiv \sum_i p_i m_i(\tau) = \sum_i p_i \theta_i (\tau + v_i)^2 = \tilde{\theta}(\tau + \tilde{v})^2 + \varphi,$$

where

$$\tilde{\theta} = \sum_i p_i \theta_i, \quad \tilde{v} = \frac{\sum_i p_i \theta_i v_i}{\sum_i p_i \theta_i}, \quad \text{and}$$

$$\varphi = \sum_i p_i \theta_i v_i^2 - \frac{(\sum_i p_i \theta_i v_i)^2}{\sum_i p_i \theta_i}. \tag{15}$$

We name $\tilde{\theta}$ and $\tilde{v}$ the *consolidated scale* parameter and the *consolidated urgency* parameter, respectively. The consolidated scale parameter ($\tilde{\theta}$) is simply the expected value of the scale parameter with respect to market uncertainly. The consolidated urgency parameter ($\tilde{v}$) is the weighted average of the individual urgency parameters, with the weight being $p_i\theta_i$ for

each v_i. Depending on the source of market uncertainty, there are two special cases:

I. *The urgency parameter is known and the uncertainty comes only from the scale parameter*. This may occur if, for instance, a firm is certain about its competitors' market behavior (represented by a constant ν) but unsure about the market size. Using Equation (15), the consolidated parameters reduce to

$$\tilde{\theta} = \sum_i p_i \theta_i, \quad \tilde{v} = v.$$

II. *The scale parameter is known and the uncertainty comes only from the urgency parameter*. This may occur if, for instance, a firm is certain about the market size (denoted by θ) but unsure about its competitors' market behavior. Under this scenario, the consolidated parameters reduce to

$$\tilde{\theta} = \theta, \quad \tilde{v} = \sum_i p_i v_i.$$

Because it is the rate of increase in the market opportunity cost (instead of its absolute value) that determines the optimal release and testing stop times, to gain further insights we examine the derivative of the expected market opportunity cost with respect to the release time. Using Equation (15), we have

$$E[M(\tau)]'_\tau = 2\tilde{\theta}\tau + 2\tilde{\theta}\tilde{v}.$$

Therefore, if the consolidated scale parameter and the consolidated urgency parameter increase by the same proportion, the increase in the scale parameter will lead to a larger impact on the Bayes solution. Hence, all else being equal, investing in obtaining a better estimate of the scale parameter could be more beneficial than estimating the urgency parameter.

We next illustrate how to obtain the Bayes solution. Based on (15), the expected total cost equals

$$E_{M(t)}[C_{\mathrm{PT}}(\tau, T)] = bN + kT + \frac{cNr}{r+1} e^{-\lambda\tau} + \frac{cN}{r+1} e^{r\lambda\tau - (r+1)\lambda T} + \tilde{\theta}(\tau + \tilde{v})^2 + \varphi.$$

Because ϕ is a constant, it does not affect the optimal release time or the testing stop time. Therefore, we can derive the solution to the PT policy using the two consolidated parameters $\tilde{\theta}$ and $\tilde{v}$. Based on Equations (13) and (14), we have

$$\tau^B_{\mathrm{PT}} = -\tilde{v} - \frac{kr}{2\tilde{\theta}(1+r)} + \frac{1}{\lambda}\mathrm{Productlog}\left(\frac{c\lambda^2 Nr}{2\tilde{\theta}(1+r)} e^{(kr+2\tilde{\theta}\tilde{v}+2\tilde{\theta}\tilde{v}r)\lambda/(2\tilde{\theta}(1+r))}\right), \quad (16)$$

$$T^B_{\mathrm{PT}} = \frac{1}{(r+1)\lambda}\ln\frac{\lambda cN}{k} + \frac{r}{r+1}\tau^B. \quad (17)$$

To illustrate, suppose the scale parameter and the urgency parameter each has three possible values, leading to the nine possible market opportunity cost functions shown in Table 3. The probabilities associated with the different parameter values are also shown in the table. The other model parameters are set to the same values as in the initial numerical example in §7. From (15), we have $\tilde{\theta} = 3{,}200$ and $\tilde{v} = 5.6$. Based on (16), and (17), we obtain the Bayes solution as $\tau^B_{\mathrm{PT}} = 23.43$ and $T^B_{\mathrm{PT}} = 130.78$. The expected total cost $C^B_{\mathrm{PT}} \equiv E_{M(t)}[C_{\mathrm{PT}}(\tau^B_{\mathrm{PT}}, T^B_{\mathrm{PT}})] = \$11.42$ million.

8.3. Risk Analysis of Bayes Solutions to the PT Policy and NPT Policy

Assuming the same market uncertainty, the solution to the NPT policy is $\tau^B_{\mathrm{NPT}} = 35.02$ days based on the Bayes risk principle. The expected cost associated with this solution is \$13.88 million. Hence, by adopting the Bayes solution for the PT policy instead that for the NPT policy, the firm can expect a cost saving of 35.6%. To further compare the Bayes solutions for the two policies, we compute the total costs associated with the policies under each of the possible market opportunity costs. The results are summarized in Table 4. For instance, the first row shows that if the true market opportunity cost is $m_1(t)$, the solution for the PT policy leads to a total cost of \$6.86 M, and the total cost under the NPT policy is \$9.94 M; the percentage difference between the two costs is 31.0%. From this table, we see that the PT policy remains far superior to the NPT policy for each of the possible market opportunity costs, with cost reductions ranging from 30.8% to 38.7%. We have also compared the two policies for other values of the scale and urgency parameters and their probabilities; the conclusions are qualitatively consistent.

Table 3 Illustrative Parameter Values and Market Opportunity Cost Functions

	Urgency parameter		
Scale parameter	$\nu = 2$ (prob. $= 0.25$)	$\nu = 6$ (prob. $= 0.40$)	$\nu = 10$ (prob. $= 0.35$)
$\theta = 1{,}000$ (prob. $= 0.30$)	$m_1(\tau) = 1{,}000(\tau+2)^2$	$m_2(\tau) = 1{,}000(\tau+6)^2$	$m_3(\tau) = 1{,}000(\tau+10)^2$
$\theta = 3{,}000$ (prob. $= 0.50$)	$m_4(\tau) = 3{,}000(\tau+2)^2$	$m_5(\tau) = 3{,}000(\tau+6)^2$	$m_6(\tau) = 3{,}000(\tau+10)^2$
$\theta = 5{,}000$ (prob. $= 0.20$)	$m_7(\tau) = 5{,}000(\tau+2)^2$	$m_8(\tau) = 5{,}000(\tau+6)^2$	$m_9(\tau) = 5{,}000(\tau+10)^2$

Table 4 Comparison Between Bayes Solutions for PT Policy and NPT Policy

	PT policy ($)	NPT policy ($)	Percentage diff. (%)
$m_1(\tau) = 1{,}000(\tau+2)^2$	6.86 M	9.94 M	31.0
$m_2(\tau) = 1{,}000(\tau+6)^2$	7.08 M	10.26 M	31.0
$m_3(\tau) = 1{,}000(\tau+10)^2$	7.33 M	10.60 M	30.8
$m_4(\tau) = 3{,}000(\tau+2)^2$	8.15 M	12.68 M	35.7
$m_5(\tau) = 3{,}000(\tau+6)^2$	8.81 M	13.62 M	35.3
$m_6(\tau) = 3{,}000(\tau+10)^2$	9.57 M	14.65 M	34.7
$m_7(\tau) = 5{,}000(\tau+2)^2$	9.45 M	15.42 M	38.7
$m_8(\tau) = 5{,}000(\tau+6)^2$	10.55 M	16.99 M	37.9
$m_9(\tau) = 5{,}000(\tau+10)^2$	11.80 M	18.71 M	36.9

9. Testing Resource Reallocation and Uncertain Market Opportunity Cost

The analyses presented so far assume that testing resources remain constant during the entire testing duration. Postrelease testing leads to several important benefits, but it also can severely strain a firm's testing resources, especially if it lasts long, as is the case for the example discussed in §7. When new projects are initiated, it may not be practical for a firm to expand its testing capabilities in the short term. In such situations, a portion of the available resources may have to be reallocated to other projects. Similarly, if other projects are completed ahead of schedule, it may be possible to allocate testing resources that become free to the focal project. In this section, we examine the PT policy in the presence of such testing resource reallocations. We consider the impact of scheduled as well as unscheduled changes in resource allocation for the focal project. To avoid overly complicating the problem, we assume that once testing stops, it is not resumed. Furthermore, we assume that the release of a new system cannot be reversed; i.e., once users start using the new system, they cannot return to an earlier system. In addition, we continue to assume that the uncertainty about market opportunity cost still exists.

9.1. Scheduled Resource Reallocation

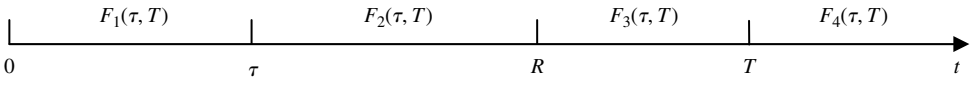
If it is known beforehand that some of the testing resources will be reallocated at a future time, the optimal release and testing stop times should still be jointly determined to minimize the cost associated with these decisions. Suppose that testing resource reallocation occurs at time R. As a result, the cost of testers' testing per unit of time and the bug failure rate due to testers' testing both change after R. For instance, if a portion of the testers is assigned to other projects, then the values of the two parameters will both decrease. We denote the two parameters before R by k and λ, and those after R by $\tilde{k}$ and $\tilde{\lambda}$. Similarly, we represent the Bayes solution without testing resource reallocation by τ^B_{PT} and T^B_{PT} and the solution with testing resource reallocation considered by $\tilde{\tau}^B_{\mathrm{PT}}$ and $\tilde{T}^B_{\mathrm{PT}}$. Although τ^B_{PT} and T^B_{PT} can be determined as described in §8, we need to develop new methods to compute $\tilde{\tau}^B_{\mathrm{PT}}$ and $\tilde{T}^B_{\mathrm{PT}}$.

When testing resources are being reallocated, three different scenarios need to be considered. First, if $\tau^B_{\mathrm{PT}} \le T^B_{\mathrm{PT}} < R$, testing resource reallocation has no effect on the new optimal solution, hence $\tilde{\tau}^B_{\mathrm{PT}} = \tau^B_{\mathrm{PT}}$ and $\tilde{T}^B_{\mathrm{PT}} = T^B_{\mathrm{PT}}$. Second, if $R \le \tau^B_{\mathrm{PT}} < T^B_{\mathrm{PT}}$, we can treat R as time zero and use the Bayes risk approach to calculate $\tilde{\tau}^B_{\mathrm{PT}}$ and $\tilde{T}^B_{\mathrm{PT}}$ based on the revised parameters $\tilde{k}$ and $\tilde{\lambda}$. The third scenario, when $\tau^B_{\mathrm{PT}} < R \le T^B_{\mathrm{PT}}$, is more complex. In this case, the optimal solution with resource reallocation considered could either result in $\tilde{\tau}^B_{\mathrm{PT}} < R \le \tilde{T}^B_{\mathrm{PT}}$ or $R \le \tilde{\tau}^B_{\mathrm{PT}} < \tilde{T}^B_{\mathrm{PT}}$. Therefore, we need to obtain two separate *candidate solutions*, one satisfying $\tilde{\tau}^B_{\mathrm{PT}} < R \le \tilde{T}^B_{\mathrm{PT}}$ and the other satisfying $R \le \tilde{\tau}^B_{\mathrm{PT}} < \tilde{T}^B_{\mathrm{PT}}$, and select the solution with the lower cost. Again, the candidate solution satisfying $R \le \tilde{\tau}^B_{\mathrm{PT}} < \tilde{T}^B_{\mathrm{PT}}$ can be computed using the Bayes risk approach by treating R as time zero. However, obtaining the candidate solution satisfying $\tilde{\tau}^B_{\mathrm{PT}} < R \le \tilde{T}^B_{\mathrm{PT}}$ requires additional analysis because the cost expression in Equation (5) is no longer valid. We next derive the expected cost associated with the scenario where $\tilde{\tau}^B_{\mathrm{PT}} < R \le \tilde{T}^B_{\mathrm{PT}}$.

The time horizon is now divided into four intervals, as shown in Figure 11. In contrast to the scenario without resource reallocation, here the interactions between testers' and users' testing processes change during postrelease testing. Therefore, testing before and after R needs to be examined separately. Following the same logic as in §4.2, we obtain the probability of a bug being detected in each of these intervals:

$$\begin{cases} F_1(\tau, T) = 1 - e^{-\lambda\tau}, \\ F_2(\tau, T) = e^{-\lambda\tau}(1 - e^{-(r+1)\lambda(R-\tau)}), \\ F_3(\tau, T) = e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)}(1 - e^{-(r\lambda+\tilde{\lambda})(T-R)}), \\ F_4(\tau, T) = e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\tilde{\lambda})(T-R)}. \end{cases} \tag{18}$$

Figure 11 Release Before R and Testing Stops After R



For each bug detected during the second time interval $[\tau, R]$, the probability that the bug is detected by users is $r/(1+r)$; and for each bug detected during the third time interval, the probability that it is detected by users is $r\lambda/(r\lambda+\tilde{\lambda})$. Therefore, the total cost associated with the PT policy equals

$$C_{\mathrm{PT}}(\tau, T) = bN + kR + \tilde{k}(T-R) + cN\left[\frac{r}{r+1}F_2(\tau, T) + \frac{r\lambda}{r\lambda+\tilde{\lambda}}F_3(\tau, T) + F_4(\tau, T)\right] + M(\tau),$$

where $M(t)$ still denotes the uncertainty in market opportunity cost. Similarly, we let $m_i(t)$ represent the market opportunity cost under market scenario i, and p_i the probability associated with market scenario i. The total expected cost, with respect to uncertain market opportunity cost, takes the following form:

$$E_{M(t)}[C_{\mathrm{PT}}(\tau, T)] = bN + kR + \tilde{k}(T-R) + cN\left[\frac{r}{r+1}F_2(\tau, T) + \frac{r\lambda}{r\lambda+\tilde{\lambda}}F_3(\tau, T) + F_4(\tau, T)\right] + \sum_i p_i m_i(\tau). \quad (19)$$

By analyzing the cost expression in (19), we find that the testing stop rule remains valid under this scenario as well.

Proposition 8. *In the presence of testing resource reallocation and uncertain market opportunity cost, it is optimal to continue testing after the reallocation until the expected number of undetected bugs reaches* $\tilde{k}/(\tilde{\lambda}c)$.

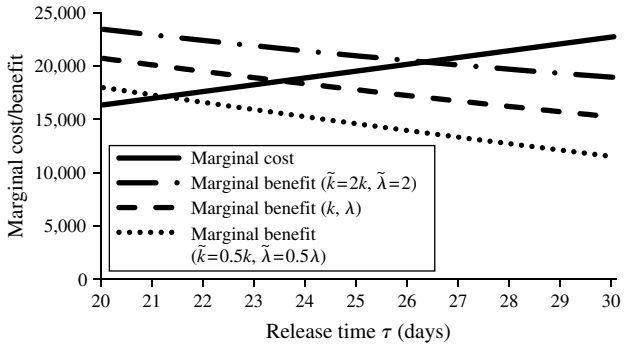
This new testing stop rule can be explained as follows. Regardless of whether the release time is optimal for the decision or not, the optimal testing stop time is computed based on the trade-off between the marginal cost of postrelease testing and the marginal benefit of postrelease testing after R. The marginal cost of postrelease testing equals $\tilde{k}$. The marginal benefit of postrelease testing equals the marginal decrease in the expected consequence of software failures in the field and is proportional to the expected number of undetected bugs at any point in time. Because the number of undetected bugs decreases over time, the marginal benefit also keeps decreasing. Therefore, testing should stop once the marginal benefit of testing equals the marginal cost of testing. If the number of undetected bugs at a given point in time is x, then the marginal benefit of testing is $c\tilde{\lambda}x$, because the instantaneous bug-detection rate for testers is $\tilde{\lambda}x$. Letting $c\tilde{\lambda}x = \tilde{k}$, we have $x = \tilde{k}/(\tilde{\lambda}c)$. Therefore, testing should stop when the expected number of undetected bugs drops to $\tilde{k}/(\tilde{\lambda}c)$. The uncertain market opportunity cost does not play a role in this trade-off because the software would have already been released by then.

A closed-form solution cannot be derived based on the cost expression in (19), but the optimal release time and optimal testing stop time can be obtained numerically. Assuming a reasonable granularity for the release time τ between 0 and R, for every value of τ, we calculate the corresponding testing stop time T based on the testing stop rule specified in Proposition 8 and record the associated cost. The optimal solution is the pair $\{\tau, T\}$ that leads to the lowest expected cost. The method is efficient because we only need to search a one-dimensional feasible region $[0, R)$ for τ.

To illustrate the PT policy with scheduled resource reallocation, we adopt the parameter values for the example discussed in §7: $N = 497$, $\lambda = 0.0308$, $k =$ \$500, $b =$ \$200, $c =$ \$50,000, and $r = 1$. The uncertain market opportunity cost has nine possible functional forms (and associated probabilities), as shown in Table 3. We first calculate the Bayes solution, assuming that resource reallocation does not occur, and obtain $\tau^B_{\mathrm{PT}} = 23.43$, $T^B_{\mathrm{PT}} = 130.78$, with an associated cost of $C^B_{\mathrm{PT}} =$ \$11.42 M. Now suppose that 50% of the testing resources (e.g., testing personnel) is scheduled to be removed from the current project at $R =$ 35. Assuming that the cost and the abilities of all testers are similar, the cost of testing per unit of time will drop to $\tilde{k} =$ \$250, and the bug failure rate due to testers' testing will reduce to $\tilde{\lambda} = 0.0154$ after R. Because $\tau^B_{\mathrm{PT}} < R \le T^B_{\mathrm{PT}}$, we first use the cost expression in (19) to determine the revised optimal testing stop time and obtain $\tilde{\tau}^B_{\mathrm{PT}} = 21.14$, $\tilde{T}^B_{\mathrm{PT}} = 161.17$, and $\tilde{C}^B_{\mathrm{PT}} =$ \$12.35 M. This candidate solution is found to be better than the one satisfying $R \le \tilde{\tau}^B_{\mathrm{PT}} < \tilde{T}^B_{\mathrm{PT}}$ and hence is the optimal solution. By comparing this solution with the one for the scenario with constant testing resources, we find that, as expected, if testing resources are to be reduced in the future, testing lasts longer and the total cost increases because users detect a higher proportion of the bugs during postrelease testing. In addition, one may expect that because testers will detect fewer bugs during postrelease testing, release should be delayed to reduce the risk to users. Surprisingly, we find that the software should be released even earlier when resources are scheduled to be reduced.

To understand this counterintuitive finding, we examine the changes in the marginal cost and the marginal benefit with the release time τ for two different reallocation levels and compare them with the original allocation. The testing stop time T is chosen such that the total cost is minimized for each given τ. The results are shown in Figure 12. In the figure, the marginal benefit represents the marginal decrease in the expected cost of software failures in

Figure 12 Marginal Benefits Affected by Testing Resource Reallocation



the field. At all three resource levels, the marginal benefit decreases monotonically as the release time is further delayed. The marginal cost in the figure equals the sum of the marginal market opportunity cost and the marginal cost of testing and is practically the same at all three resource levels; therefore, we show only one marginal cost curve in the figure. As expected, the marginal cost increases with the release time primarily because of the convexity of the market opportunity cost. The software should be released when the marginal cost equals the marginal benefit. From Figure 12, we see that the marginal benefit is lower with reduced testing resources ($\tilde{k} = 0.5k$, $\tilde{\lambda} = 0.5\lambda$) after reallocation. This is because testers' instantaneous bug-detection rate decreases with fewer resources. As shown in the figure, the lower marginal benefit curve intersects the marginal cost curve at an earlier release time. Similarly, the marginal benefit and marginal cost curves meet at a later release time with increased testing resources ($\tilde{k} = 2k$, $\tilde{\lambda} = 2\lambda$) after reallocation. This explains why it is optimal for the firm to release the software earlier when testing resources are reduced after reallocation and to further delay the release when resources are increased after reallocation.

9.2. Unscheduled Testing Reallocation

We next examine the scenario where testing resource reallocation occurs unexpectedly. If the reallocation is made before release, the method to cope with it is straightforward. We treat the time of change as time zero and recompute the optimal release time and testing stop time based on the revised cost of testing and bug failure rate due to testing. However, if the reallocation occurs after release, we need to recompute only the testing stop time. The cost formulation (19) and the testing stop rule described in Proposition 8 remain valid under this scenario.

We denote the expected number of undetected bugs at R by $u(R)$, which equals $Ne^{-\lambda\tau}e^{-(r+1)\lambda(R-\tau)}$, where τ is the release time. The revised optimal testing stop time takes the following value:

$$\tilde{T}^B_{\mathrm{PT}} = \begin{cases} R, & \text{if } u(R) \le \tilde{k}/(\tilde{\lambda}c), \\ R + \ln\left(\dfrac{\tilde{\lambda}cu(R)}{\tilde{k}}\right)\Big/(r\lambda + \tilde{\lambda}), & \\ & \text{if } u(R) > \tilde{k}/(\tilde{\lambda}c). \end{cases} \tag{20}$$

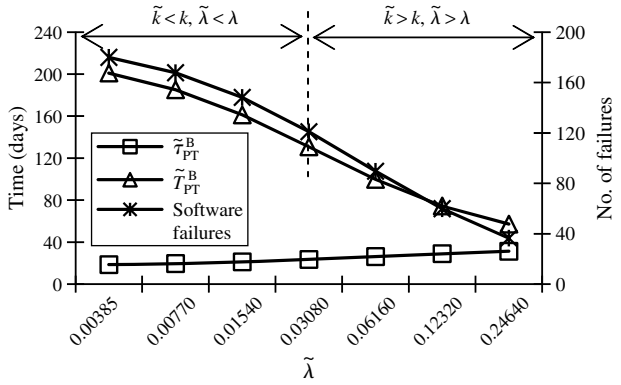
The solution shown in (20) follows from the testing stop rule: If the expected number of undetected bugs at time R already satisfies the testing stop rule after R, then testing should stop at R; otherwise, testing should continue until the rule is satisfied. Again, because the software has already been released, the uncertain market opportunity cost does not affect the optimal testing stop time.

9.3. Impact of Testers' Cost Effectiveness on the Release Policy

Because testing resource reallocation does not change the consequence of a software failure in the field (c), the testing stop condition is determined solely by the testers' *cost effectiveness*, i.e., the ratio between the cost of testing per unit of time and the bug failure rate due to testers' testing (k/λ or $\tilde{k}/\tilde{\lambda}$). Depending on whether this ratio increases, decreases, or remains constant after reallocation, the impact of testing resource reallocation on the testing stop decision and the risk to users are different.

If testers' cost effectiveness remains the same after resource reallocation, i.e., $\tilde{k}/\tilde{\lambda} = k/\lambda$, the expected number of undetected bugs at the optimal testing stop time does not change after testing resource reallocation. In this case, if the last phase of testing is conducted using reduced testing resources, because testers' bug-detection rate decreases, testing will last longer. Further, because testers are expected to detect a smaller proportion of bugs after R, the expected cost of software failures in the field will increase. In contrast, if the last phase of testing is conducted using more testing resources, testing will be shortened and testers will detect a larger proportion of the bugs, and hence the expected cost of software failures in the field will decrease. To better understand this impact, we vary the values of $\tilde{k}$ and $\tilde{\lambda}$ while keeping their ratio fixed at the original level (k/λ) and repeat the numerical analysis. The solutions are summarized in Figure 13. The first two data points in the figure represent reduced testing resources after reallocation, and the last two represent increased resources after reallocation. From this figure, we can see that as the testing resources devoted to the focal project increase after reallocation, the optimal testing stop time and the expected number of software failures in the field both decrease as expected. However, the optimal release

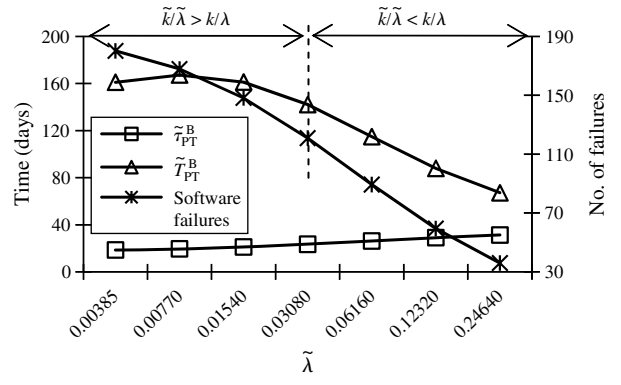
Figure 13 **Impact of the Amount of Testing Resources after Reallocation** ($\tilde{k}/\tilde{\lambda} = k/\lambda$)



time is delayed. This is consistent with the release time phenomenon discussed in §9.1.

During testing resource reallocation, if the more experienced and productive testers are assigned to other projects, the remaining testers' bug-detection effectiveness may decrease proportionately more than the cost of testers' testing, thus leading to $\tilde{k}/\tilde{\lambda} > k/\lambda$. This implies that testers' testing becomes more expensive relative to their testing effectiveness. Based on Proposition 8, more bugs will remain undetected after testing stops. Yet assigning the less productive testers to other projects can result in $\tilde{k}/\tilde{\lambda} < k/\lambda$. This implies that the remaining testers are less expensive relative to testers' effectiveness; hence fewer bugs will remain after testing stops. Similar arguments follow if testing resources freed from other projects are added to the current project. To further examine how the testers' cost effectiveness impacts the solution to the PT policy and the risk to users, we fix the cost of testing at $\tilde{k} = \$50$, representing a decrease in testing resources after reallocation, and vary $\tilde{\lambda}$ from 0.00385 to 0.2464; the solutions are shown in Figure 14. From the figure, we see that as testers become more effective relative to their cost, it is optimal to delay the release time, the optimal testing stop time first increases and then decreases, and the expected number of software failures in the field decreases monotonically. Analyses conducted for scenarios with increased testing resources lead to similar results. The delay in release time follows from the phenomenon discussed earlier. The finding regarding the decreasing number of software failures in the field is as expected. The nonmonotonic result regarding the testing stop time with increasing $\tilde{\lambda}$ is similar to the impact of testing stop time due to increasing λ as discussed in §7 (Figure 10).

Figure 14 **Impact of Testers' Bug-Detection Effectiveness with Reduced Resources** ($\tilde{k} = 500$)



10. Contributions, Managerial Implications, and Future Research Directions

Traditionally, firms may have had to delay the release of their software to ensure reliability; however, this could result in substantial market opportunity costs. We propose a novel software release policy with postrelease testing that helps mitigate this issue. Specifically, we (i) formally analyze the bug-detection behavior under such a policy, accounting for periods when bugs are detected just by testers, as well as by both testers and users; (ii) develop a model that can be used to determine the cost-minimizing release and testing stop times; and (iii) analytically show the advantages of the PT policy over the NPT policy. We find that the software should be released earlier and testing should stop later under the proposed policy than under the NPT policy. Interestingly, we also find that although the expected number of undetected bugs is higher at the time of release, the expected number of software failures in the field is reduced under the proposed policy. We extend our model to study the release policy under more complex scenarios involving uncertain market opportunity costs and testing resource reallocations. We show that all our prior findings remain valid under uncertain market opportunity costs. Surprisingly, we find that the software should be released earlier when testing resources are to be reduced after release.

The findings of this study have important implications for managing software projects, which have a notorious track record for falling behind schedule and/or over budget (Standish Group International 2001). Our analyses show that when opportunity costs are significant, by delinking the testing stop time from the release time, the development team can deliver the system earlier and simultaneously reduce both the risk of software failures in the field over the lifetime of the product and the overall cost through extended testing after release. Therefore, if the release and testing stop times are determined based on the proposed policy, project managers need not be concerned with an early-release decision, because the risk will be

more than offset by the benefits of postrelease testing. This additional flexibility can help significantly accelerate the release time and help firms gain an advantageous position in a competitive marketplace. Furthermore, with existing release policies, new systems can be significantly "undertested," because project managers fear the consequences of delaying release. Our release policy, in contrast, allows firms to invest more on software testing, eventually leading to a more reliable system. Besides reducing the chances of failure in the field, a more reliable system helps the firm reap intangible benefits such as increased user satisfaction and customer loyalty. Last, our policy allows a firm to better utilize its testing resources. If testing stops immediately after a new system is put into operation, testing resources are often underutilized or even completely idle between projects. If this is the case, firms can test even longer to further improve the quality of the system, because the marginal cost of testing is relatively low.

The policy with postrelease testing can also be modified to incorporate the effect of investment in learning. Assuming that the effect of such investments can be captured, our model can be extended to help decide the optimal amount of investment in learning as well as the optimal release and testing stop times. In a preliminary analysis, with the simplifying assumption that to sustain the effect of learning the investment in learning has to be continuous, we find that it is optimal to release the software later with greater investment in learning or when the investment in learning results in a higher rate of learning. In addition, we find that when there are fewer testing resources remaining after reallocation, the firm is better off releasing the software earlier. Future research could look at a more general approach to learning, where one could model the cumulative benefit of investment in learning over time. Possible solution approaches such as dynamic programming techniques could be used to determine the optimal amount of investment in learning over time.

There are a number of other research directions that merit further consideration. First, we currently assume that the number of bugs and the failure rates are known beforehand. It should be feasible to develop multiperiod decision models that help organizations make better decisions based on software quality information collected until each decision time. Second, we consider only custom-built enterprise-level information systems in this study. The desirable duration of public beta testing for commercial off-the-shelf software could also be analyzed based on the methodology we propose in this study. Third, the cost of installing patches is assumed to be negligible in this research, because it is typically significantly smaller than the other costs we consider. For those cases where the cost of patching is significant, future research could model both postrelease testing and patch management to minimize the total cost to a firm. Fourth, when analyzing unscheduled testing resource reallocation, the time of reallocation (R) is assumed to be known once the reallocation decision is made. Future research could treat R as a random variable and reexamine the decision problem. Finally, future study could also consider multiple resource reallocations during the testing period.

Appendix

A.1. Derivation of Equation (4)

Suppose X_1 and X_2 are two independent exponential random variables with failure rates λ_1 and λ_2, respectively. The probability that X_1 fails before X_2, given that at least one of them fails before time D, equals

$$\frac{P\{X_1 < X_2, X_2 < D\} + P\{X_1 < D, X_2 > D\}}{1 - e^{-(\lambda_1+\lambda_2)D}}.$$

We next derive the two terms shown in the numerator of the above expression.

$$\begin{aligned}
P\{X_1 < X_2, X_2 < D\} &= \int_0^D P\{X_1 < X_2 \mid X_2 = x\}\lambda_2 e^{-\lambda_2 x}\,dx \\
&= \int_0^D P\{X_1 < x\}\lambda_2 e^{-\lambda_2 x}\,dx \\
&= \int_0^D (1 - e^{-\lambda_1 x})\lambda_2 e^{-\lambda_2 x}\,dx \\
&= \int_0^D \lambda_2 e^{-\lambda_2 x}\,dx - \int_0^D e^{-\lambda_1 x}\lambda_2 e^{-\lambda_2 x}\,dx \\
&= -e^{-\lambda_2 x}|_0^D + \frac{\lambda_2}{\lambda_1+\lambda_2} e^{-(\lambda_1+\lambda_2)x}|_0^D \\
&= -e^{-\lambda_2 D} + 1 + \frac{\lambda_2}{\lambda_1+\lambda_2} e^{-(\lambda_1+\lambda_2)D} \\
&\quad - \frac{\lambda_2}{\lambda_1+\lambda_2}.
\end{aligned}$$

$$P\{X_1 < D, X_2 > D\} = (1 - e^{-\lambda_1 D})e^{-\lambda_2 D} = e^{-\lambda_2 D} - e^{-(\lambda_1+\lambda_2)D}.$$

Therefore,

$$\begin{aligned}
&\frac{P\{X_1 < X_2, X_2 < D\} + P\{X_1 < D, X_2 > D\}}{1 - e^{-(\lambda_1+\lambda_2)D}} \\
&\quad = \frac{\lambda_1/(\lambda_1+\lambda_2)[1 - e^{-(\lambda_1+\lambda_2)D}]}{1 - e^{-(\lambda_1+\lambda_2)D}} = \frac{\lambda_1}{\lambda_1+\lambda_2}.
\end{aligned}$$

A.2. Proof of Lemma 1

To prove that $C_{PT}(\tau,T)$ is a strictly convex function of τ and T, we only need to show the following:

$$\begin{cases}
\partial^2 C_{PT}/\partial\tau^2 > 0, \\
\partial^2 C_{PT}/\partial T^2 > 0, \\
(\partial^2 C_{PT}/\partial\tau^2)(\partial^2 C_{PT}/\partial T^2) - (\partial^2 C_{PT}/\partial\tau\partial T)^2 > 0.
\end{cases}$$

We first obtain the first-order derivatives:

$$\partial C_{\mathrm{PT}}/\partial\tau = -\frac{\lambda cNr}{r+1}e^{-\lambda\tau} + \frac{\lambda cNr}{r+1}e^{r\lambda\tau}e^{-(r+1)\lambda T} + m'(\tau), \quad \text{and}$$
$$\partial C_{\mathrm{PT}}/\partial T = k - \lambda cNe^{r\lambda\tau}e^{-(r+1)\lambda T}.$$

The second-order derivatives are as follows:

$$\partial^2 C_{\mathrm{PT}}/\partial\tau^2 = \frac{\lambda^2 cNr}{r+1}e^{-\lambda\tau} + \frac{r^2\lambda^2 cN}{r+1}e^{r\lambda\tau}e^{-(r+1)\lambda T} + m''(\tau) > 0, \tag{21}$$

$$\partial^2 C_{\mathrm{PT}}/\partial T^2 = (r+1)\lambda^2 cNe^{r\lambda\tau}e^{-(r+1)\lambda T} > 0, \tag{22}$$
$$\partial^2 C_{\mathrm{PT}}/\partial\tau\partial T = -\lambda^2 rcNe^{r\lambda\tau}e^{-(r+1)\lambda T}.$$

Given the second-order derivatives, we have

$$\begin{aligned}(\partial^2 C_{\mathrm{PT}}/\partial\tau^2)(\partial^2 C_{\mathrm{PT}}/\partial T^2) - (\partial^2 C_{\mathrm{PT}}/\partial\tau\partial T)^2 &= \lambda^4 c^2 N^2 r e^{(r-1)\lambda\tau}e^{-(r+1)\lambda T} \\ &\quad + m''(\tau)(r+1)\lambda^2 cNe^{r\lambda\tau}e^{-(r+1)\lambda T} > 0.\end{aligned} \tag{23}$$

From (21), (22), and (23), we conclude that $C_{\mathrm{PT}}(\tau, T)$ is a strictly convex function of τ, T. □

A.3. Proof of Proposition 1

Without the market opportunity cost, we obtain from (6) and (11) the following solution:

$$\tau^*_{\mathrm{NPT}} = \tau^*_{\mathrm{PT}} = \frac{1}{\lambda}\ln\frac{\lambda cN}{k}. \tag{24}$$

From (7), we have

$$e^{-\lambda\tau^*_{\mathrm{PT}}} = e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} \;\Rightarrow\; e^{-(r+1)\lambda\tau^*_{\mathrm{PT}}} = e^{-(r+1)\lambda T^*_{\mathrm{PT}}} \;\Rightarrow\; \tau^*_{\mathrm{PT}} = T^*_{\mathrm{PT}}.$$

Therefore,

$$\tau^*_{\mathrm{NPT}} = \tau^*_{\mathrm{PT}} = T^*_{\mathrm{PT}} = \frac{1}{\lambda}\ln\frac{\lambda cN}{k}. \tag{25}$$

The solution shown in (24) is valid only if the solution is interior, or equivalently if $k < \lambda cN$. We next consider the boundary solutions. From (24), we conclude $\tau^*_{\mathrm{NPT}} = \tau^*_{\mathrm{PT}} = 0$, iff $k \geq \lambda cN$.

Substituting τ^*_{PT} with zero in (8), we obtain

$$T^*_{\mathrm{PT}} = \frac{1}{(r+1)\lambda}\ln\left(\frac{\lambda cN}{k}\right). \tag{26}$$

With the condition $k \geq \lambda cN$, T^*_{PT} in (A6) is less than zero. Hence T^*_{PT} should also take the boundary value of zero. We therefore conclude $\tau^*_{\mathrm{NPT}} = \tau^*_{\mathrm{PT}} = T^*_{\mathrm{PT}} = 0$, iff $k \geq \lambda cN$. □

A.4. Proof of Proposition 2

(i) We first prove $\tau^*_{\mathrm{PT}} < \tau^*_{\mathrm{NPT}}$ for interior solutions. We define the following two functions:

$$f_{\mathrm{NPT}}(\tau) = -\lambda cNe^{-\lambda\tau} + m'(\tau), \quad \text{and}$$
$$f_{\mathrm{PT}}(\tau) = -\lambda cNe^{-\lambda\tau} + \frac{r+1}{r}m'(\tau).$$

From $m'(\tau) > 0$ and $m''(\tau) \geq 0$, we conclude

$$f'_{\mathrm{NPT}}(\tau) > 0, \quad f'_{\mathrm{PT}}(\tau) > 0, \quad f_{\mathrm{NPT}}(\tau) < f_{\mathrm{PT}}(\tau), \quad \forall\tau. \tag{27}$$

Substituting the two newly defined functions in (6) and (11), we have

$$k + f_{\mathrm{NPT}}(\tau^*_{\mathrm{NPT}}) = 0, \tag{28}$$
$$k + f_{\mathrm{PT}}(\tau^*_{\mathrm{PT}}) = 0. \tag{29}$$

Equations (28) and (29) lead to

$$f_{\mathrm{NPT}}(\tau^*_{\mathrm{NPT}}) = f_{\mathrm{PT}}(\tau^*_{\mathrm{PT}}). \tag{30}$$

From (27) and (30), we conclude $\tau^*_{\mathrm{PT}} < \tau^*_{\mathrm{NPT}}$.

(ii) We next prove $\tau^*_{\mathrm{NPT}} < T^*_{\mathrm{PT}}$ for interior solutions. Note that from (7), we have

$$e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} = e^{-\lambda\tau^*_{\mathrm{PT}}} - m'(\tau^*_{\mathrm{PT}})\frac{r+1}{\lambda cNr} < e^{-\lambda\tau^*_{\mathrm{PT}}}$$
$$\Rightarrow\; e^{-(r+1)\lambda T^*_{\mathrm{PT}}} < e^{-(\lambda+1)\tau^*_{\mathrm{PT}}} \;\Rightarrow\; T^*_{\mathrm{PT}} > \tau^*_{\mathrm{PT}}.$$

From (6) and (8), we obtain

$$\begin{aligned}k &= \lambda cNe^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} \\ &= \lambda cNe^{-\lambda\tau^*_{\mathrm{NPT}}} - m'(\tau^*_{\mathrm{NPT}}), \quad \text{or equivalently,} \\ e^{-\lambda\tau^*_{\mathrm{NPT}}} - e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} &= m'(\tau^*_{\mathrm{NPT}})/(\lambda cN).\end{aligned} \tag{31}$$

From (7), we also have

$$\frac{r}{r+1}(e^{-\lambda\tau^*_{\mathrm{PT}}} - e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}}) = m'(\tau^*_{\mathrm{PT}})/(\lambda cN). \tag{32}$$

From $\tau^*_{\mathrm{PT}} < \tau^*_{\mathrm{NPT}}$ and $m'(\tau) > 0$, we conclude that the right-hand side of (31) is greater than the right-hand side of (32); therefore, we must have

$$\begin{aligned}&e^{-\lambda\tau^*_{\mathrm{NPT}}} - e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} > \frac{r}{r+1}(e^{-\lambda\tau^*_{\mathrm{PT}}} - e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}}), \\ &\Leftrightarrow\; e^{-\lambda\tau^*_{\mathrm{NPT}}} > \frac{r}{r+1}e^{-\lambda\tau^*_{\mathrm{PT}}} + \frac{1}{r+1}e^{r\lambda\tau^*_{\mathrm{PT}}}e^{-(r+1)\lambda T^*_{\mathrm{PT}}} \\ &\Leftrightarrow\; e^{-\lambda\tau^*_{\mathrm{NPT}}} > e^{-\lambda T^*_{\mathrm{PT}}}\left(\frac{r}{r+1}e^{\lambda(T^*_{\mathrm{PT}}-\tau^*_{\mathrm{PT}})} + \frac{1}{r+1}e^{-r\lambda(T^*_{\mathrm{PT}}-\tau^*_{\mathrm{PT}})}\right) \\ &\Leftrightarrow\; e^{-\lambda\tau^*_{\mathrm{NPT}}} > e^{-\lambda T^*_{\mathrm{PT}}}\left(\frac{r}{r+1}x + \frac{1}{r+1}x^{-r}\right),\end{aligned} \tag{33}$$

where $x = e^{\lambda(T^*_{\mathrm{PT}}-\tau^*_{\mathrm{PT}})} > 1$ because $T^*_{\mathrm{PT}} > \tau^*_{\mathrm{PT}}$.

If we let $f(x) = r/(r+1)x + 1(r+1)x^{-r}$, then we have $f(1) = 1$ and

$$f'(x) = \frac{r}{r+1}(1 - x^{-r-1}) > 0, \quad \forall x > 1.$$

Therefore,

$$f(x) = \frac{r}{r+1}x + \frac{1}{r+1}x^{-r} > 1, \quad \forall x > 1. \tag{34}$$

(33) and (34) jointly lead to

$$e^{-\lambda\tau^*_{\mathrm{NPT}}} > e^{-\lambda T^*_{\mathrm{PT}}} \;\Rightarrow\; \tau^*_{\mathrm{NPT}} < T^*_{\mathrm{PT}}.$$

(iii) We now examine the boundary solutions for the NPT policy and the PT policy. For both policies, boundary solutions should be used if any of the values obtained from the first-order conditions (6), (8), and (11) is less than zero. We first derive the conditions under which boundary solutions should be used.

Because $C_{\mathrm{NPT}}(\tau)$ is strictly convex in τ, $\tau^*_{\mathrm{NPT}} = 0$ if and only if $C'_{\mathrm{NPT}}(0) = k - \lambda cN + m'(0) \geq 0$, or

$$\lambda cN \leq k + m'(0). \tag{35}$$

For $C_{PT}(\tau, T)$, we conclude from (11) that $\tau^*_{PT} = 0$ if and only if

$$\lambda cN \le k + \frac{r+1}{r} m'(0). \tag{36}$$

Obviously, if (35) holds, (36) must also be true. We therefore conclude that if τ^*_{NPT} takes the boundary solution, τ^*_{PT} must also be at the boundary, i.e., $\tau^*_{NPT} = 0 \Rightarrow \tau^*_{PT} = 0$.

When τ^*_{PT} equals zero, to decide the optimal solution for T^*_{PT}, we define

$$C_{PT0}(T) \equiv C_{PT}(0, T) = bN + kT + \frac{cNr}{r+1} + \frac{cN}{r+1} e^{-(r+1)\lambda T} + m(0).$$

If we allow T^*_{PT} to take any real value, from the first-order condition of $C_{PT0}(T)$, we have

$$k - \lambda cNe^{-(r+1)\lambda T^*_{PT}} = 0 \;\Rightarrow\; T^*_{PT} = \frac{1}{(r+1)\lambda} \ln\left(\frac{\lambda cN}{k}\right). \tag{37}$$

We therefore conclude that $T^*_{PT} = 0$ if and only if

$$\lambda cN \le k. \tag{38}$$

Clearly, if (38) is satisfied, (35) and (36) must also hold. Therefore,

$$T^*_{PT} = 0 \;\Rightarrow\; (\tau^*_{NPT} = 0 \text{ AND } \tau^*_{PT} = 0).$$

The following summarizes the boundary solution scenarios and their corresponding conditions:

(a) When $\lambda cN \le k$, $\tau^*_{PT} = \tau^*_{NPT} = T^*_{PT} = 0$,

(b) When $k < \lambda cN \le k + m'(0)$, $\tau^*_{PT} = \tau^*_{NPT} = 0$, $T^*_{PT} = (1/((r+1)\lambda)) \ln(\lambda cN/k) > 0$, and

(c) When $k + m'(0) < \lambda cN \le k + ((r+1)/r)m'(0)$, $\tau^*_{PT} = 0$, $\tau^*_{NPT} > 0$, $T^*_{PT} = (1/((r+1)\lambda)) \ln(\lambda cN/k) > 0$.

The first two scenarios clearly satisfy $\tau^*_{PT} \le \tau^*_{NPT} \le T^*_{PT}$. We next prove that $\tau^*_{NPT} < T^*_{PT}$ holds for the third scenario. With the solution values shown in (3), the following equality is still valid:

$$\lambda cNe^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}} = k. \tag{39}$$

For ease of comparison, we denote the optimal solutions obtained from the original first-order conditions (8) and (11) by τ^0_{PT} (here $\tau^0_{PT} \le 0 = \tau^*_{PT}$) and T^0_{PT}. Therefore, τ^0_{PT} and T^0_{PT} also satisfy

$$\lambda cNe^{r\lambda\tau^0_{PT}} e^{-(r+1)\lambda T^0_{PT}} = k. \tag{40}$$

Because τ^*_{NPT}, τ^0_{PT}, and T^0_{PT} are obtained from the original first-order conditions (6), (8), and (11), from Part (b) of the proof of Proposition 2 we have

$$\tau^*_{NPT} < T^0_{PT}. \tag{41}$$

From (39) and (40), we conclude

$$e^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}} = e^{r\lambda\tau^0_{PT}} e^{-(r+1)\lambda T^0_{PT}}. \tag{42}$$

(42) and $\tau^0_{PT} \le 0 = \tau^*_{PT}$ jointly lead to

$$T^*_{PT} \ge T^0_{PT}. \tag{43}$$

From (43) and (41), we conclude $\tau^*_{NPT} < T^*_{PT}$. □

A.5. Proof of Proposition 3

Based on $F_3(\tau, T)$ derived in §4.2, the expected number of undetected bugs at the optimal testing stop time, denoted by $u(T^*_{PT})$, equals $Ne^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}}$. From the first-order condition (8), we have

$$u(T^*_{PT}) = Ne^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}} = k/(\lambda c).$$

Therefore, the stopping rule is satisfied when the optimal solution is interior.

Similarly, it can be verified that the three boundary solutions scenarios shown in Table 2 all satisfy

$$u(T^*_{PT}) \le k/(\lambda c). \quad \square$$

A.6. Proof of Proposition 6

The expected number of software failures in the field under the NPT policy is

$$S_{NPT} = Ne^{-\lambda\tau^*_{NPT}}. \tag{44}$$

The expected number of software failures in the field under the PT policy equals

$$\begin{aligned} S_{PT} &= N(F_2(\tau^*_{PT}, T^*_{PT}) \frac{r}{r+1} + F_3(\tau^*_{PT}, T^*_{PT})) \\ &= \frac{r}{r+1} Ne^{-\lambda\tau^*_{PT}} + \frac{1}{r+1} Ne^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}}. \end{aligned} \tag{45}$$

As shown in Table 2, we need to compare S_{NPT} and S_{PT} under three different solution scenarios: (i) $T^*_{PT} > \tau^*_{NPT} > \tau^*_{PT} > 0$; (ii) $\tau^*_{PT} = 0$, $T^*_{PT} > \tau^*_{NPT} > 0$, and (iii) $\tau^*_{PT} = 0$, $\tau^*_{NPT} = 0$, $T^*_{PT} > 0$. The third scenario is straightforward. Therefore we focus on scenarios (i) and (ii). We first consider the strictly convex market opportunity cost function, i.e., $m''(\tau) > 0$.

(i) Under the $T^*_{PT} > \tau^*_{NPT} > \tau^*_{PT} > 0$ scenario, all solutions are interior. Based on (6), (8), and (11), we have

$$Ne^{-\lambda\tau^*_{NPT}} = \frac{k + m'(\tau^*_{NPT})}{\lambda c}, \tag{46}$$

$$Ne^{r\lambda\tau^*_{PT}} e^{-(r+1)\lambda T^*_{PT}} = \frac{k}{\lambda c}, \tag{47}$$

$$Ne^{-\lambda\tau^*_{PT}} = \frac{k + ((r+1)/r)m'(\tau^*_{PT})}{\lambda c}. \tag{48}$$

From (44) and (46), we obtain

$$S_{NPT} = \frac{k + m'(\tau^*_{NPT})}{\lambda c}. \tag{49}$$

Substituting (47) and (48) in (45) yields

$$\begin{aligned} S_{PT} &= \frac{r}{r+1} \frac{k + ((r+1)/r)m'(\tau^*_{PT})}{\lambda c} + \frac{1}{r+1} \frac{k}{\lambda c} \\ &= \frac{k + m'(\tau^*_{PT})}{\lambda c}. \end{aligned} \tag{50}$$

From $\tau^*_{NPT} > \tau^*_{PT}$ and $m''(\tau) > 0$, we have

$$m'(\tau^*_{NPT}) > m'(\tau^*_{PT}). \tag{51}$$

From (49), (50), and (51), we conclude $S_{PT} < S_{NPT}$.

(ii) For the $\tau^*_{PT}=0$, $T^*_{PT}>\tau^*_{NPT}>0$ scenario, only τ^*_{PT} takes a boundary solution. The expression shown in (49) for the NPT policy is still valid, and we have

$$S_{NPT}=\frac{k+m'(\tau^*_{NPT})}{\lambda c}>\frac{k+m'(0)}{\lambda c}=\frac{r}{r+1}\frac{k+m'(0)}{\lambda c}+\frac{1}{r+1}\frac{k+m'(0)}{\lambda c}.$$

For S_{PT}, by setting $\tau^*_{PT}=0$ and $T^*_{PT}=(1/((r+1)\lambda))\cdot\ln(\lambda cN/k)$ in (45), we obtain

$$S_{PT}=\frac{r}{r+1}N+\frac{1}{r+1}\frac{k}{\lambda c}.$$

Therefore,

$$\begin{aligned}S_{NPT}-S_{PT}&>\left(\frac{r}{r+1}\frac{k+m'(0)}{\lambda c}+\frac{1}{r+1}\frac{k+m'(0)}{\lambda c}\right)-\left(\frac{r}{r+1}N+\frac{1}{r+1}\frac{k}{\lambda c}\right)\\&=\frac{r}{r+1}\frac{k+m'(0)}{\lambda c}+\frac{1}{r+1}\frac{m'(0)}{\lambda c}-\frac{r}{r+1}N\\&=\frac{r}{(r+1)\lambda c}\left(k+\frac{r+1}{r}m'(0)-\lambda cN\right)\ge 0.\end{aligned}\tag{52}$$

The last inequality holds because of the condition $k+m'(0)<\lambda cN\le k+((r+1)/r)m'(0)$, which is required for the solutions to satisfy $\tau^*_{PT}=0$, $T^*_{PT}>\tau^*_{NPT}>0$. From (52), we conclude $S_{PT}<S_{NPT}$.

Analogously, it can be shown that with a linear market opportunity cost function, i.e., $m''(\tau)=0$, we have $S_{PT}=S_{NPT}$ for scenario (i) and $S_{PT}\le S_{NPT}$ for scenario (ii). □

A.7. Proof of Proposition 7

(i) **The impact of** c **and** N**.** We first prove the conclusion with respect to c. For expositional convenience, we let $f_{PT}(\tau)=-\lambda cNe^{-\lambda\tau}+((r+1)/r)m'(\tau)$. If the solutions are interior, we know from (11) that $k+f_{PT}(\tau^*_{PT})=0$. Now suppose that the parameter c is changed to $\tilde c>c$. To reflect this change, we define a new function:

$$\tilde f_{PT}(\tau)=-\lambda\tilde cNe^{-\lambda\tau}+\frac{r+1}{r}m'(\tau).\tag{53}$$

These two functions have the following property:

$$\tilde f'_{PT}(\tau)>0,\quad f'_{PT}(\tau)>0,\quad \tilde f_{PT}(\tau)<f_{PT}(\tau),\quad \forall\tau.\tag{54}$$

We denote the optimal solution associated with the new parameter $\tilde c$ by $\tilde\tau^*_{PT}$ and $\tilde T^*_{PT}$. If the new solution $\tilde\tau^*_{PT}$ is still interior, we must have $k+\tilde f_{PT}(\tilde\tau^*_{PT})=0$.

Because $k+f_{PT}(\tau^*_{PT})=0$ also holds, we have

$$\tilde f_{PT}(\tilde\tau^*_{PT})=f_{PT}(\tau^*_{PT}).\tag{55}$$

(54) and (55) jointly lead to

$$\tilde\tau^*_{PT}>\tau^*_{PT}.\tag{56}$$

We now examine how $(T^*_{PT}-\tau^*_{PT})$ changes with c. Let $d=T^*_{PT}-\tau^*_{PT}$ and $\tilde d=\tilde T^*_{PT}-\tilde\tau^*_{PT}$; from (8) we have

$$\begin{aligned}&\lambda\tilde cNe^{r\lambda\tilde\tau^*_{PT}}e^{-(r+1)\lambda(\tilde\tau^*_{PT}+\tilde d)}=\lambda cNe^{r\lambda\tau^*_{PT}}e^{-(r+1)\lambda(\tau^*_{PT}+d)},\quad\text{or}\\&\lambda\tilde cNe^{-\lambda\tilde\tau^*_{PT}}e^{-(r+1)\lambda\tilde d}=\lambda cNe^{-\lambda\tau^*_{PT}}e^{-(r+1)\lambda d}.\end{aligned}\tag{57}$$

Further, (55) is equivalent to

$$-\lambda\tilde cNe^{-\lambda\tilde\tau^*_{PT}}+\frac{r+1}{r}m'(\tilde\tau^*_{PT})=-\lambda cNe^{-\lambda\tau^*_{PT}}+\frac{r+1}{r}m'(\tau^*_{PT}).\tag{58}$$

We first consider a strictly convex market opportunity cost, i.e., $m''(\tau)>0$. From (56), we have

$$\frac{r+1}{r}m'(\tilde\tau^*_{PT})>\frac{r+1}{r}m'(\tau^*_{PT}).\tag{59}$$

From (58) and (59), we conclude

$$\lambda\tilde cNe^{-\lambda\tilde\tau^*_{PT}}>\lambda cNe^{-\lambda\tau^*_{PT}}.\tag{60}$$

(57) and (60) together lead to

$$\begin{aligned}&e^{-(r+1)\lambda\tilde d}<e^{-(r+1)\lambda d}\ \Rightarrow\ \tilde d>d\quad\text{or}\\&(\tilde T^*_{PT}-\tilde\tau^*_{PT})>(T^*_{PT}-\tau^*_{PT}).\end{aligned}\tag{61}$$

From (56) and (61), we also conclude $\tilde T^*_{PT}>T^*_{PT}$.

We next consider a linear market opportunity cost, i.e., $m''(\tau)=0$. By reexamining (59) through (61), we conclude that $(\tilde T^*_{PT}-\tilde\tau^*_{PT})=(T^*_{PT}-\tau^*_{PT})$ and $\tilde T^*_{PT}>T^*_{PT}$ hold when $m''(\tau)=0$.

The conclusions with respect to N can be proved analogously.

(ii) **The impact of** k**.** Assume k increases to $\tilde k>k$. We denote the new optimal solutions by $\tilde\tau^*_{PT}$ and $\tilde T^*_{PT}$. If all solutions are interior, from (11) and (29), we have

$$\begin{aligned}&\tilde k-\lambda cNe^{-\lambda\tilde\tau^*_{PT}}+\frac{r+1}{r}m'(\tilde\tau^*_{PT})=k-\lambda cNe^{-\lambda\tau^*_{PT}}+\frac{r+1}{r}m'(\tau^*_{PT}),\end{aligned}\tag{62}$$

$$\tilde k+f_{PT}(\tilde\tau^*_{PT})=k+f_{PT}(\tau^*_{PT}).\tag{63}$$

Because $\tilde k>k$, and $f'_{PT}(\tau)>0$, from (63) we must have $\tilde\tau^*_{PT}<\tau^*_{PT}$.

From $m''(\tau)\ge 0$, we further conclude

$$\frac{r+1}{r}m'(\tilde\tau^*_{PT})\le\frac{r+1}{r}m'(\tau^*_{PT}).\tag{64}$$

Based on and (62) and (64), we have

$$\begin{aligned}&\tilde k-\lambda cNe^{-\lambda\tilde\tau^*_{PT}}\ge k-\lambda cNe^{-\lambda\tau^*_{PT}}\ \Leftrightarrow\ \tilde k-k\\&\qquad\ge\lambda cNe^{-\lambda\tilde\tau^*_{PT}}-\lambda cNe^{-\lambda\tau^*_{PT}}.\end{aligned}\tag{65}$$

Let $d=T^*_{PT}-\tau^*_{PT}$ and $\tilde d=\tilde T^*_{PT}-\tilde\tau^*_{PT}$. From (8), we get

$$\begin{aligned}&\tilde k-\lambda cNe^{r\lambda\tilde\tau^*_{PT}}e^{-(r+1)\lambda(\tilde\tau^*_{PT}+\tilde d)}=k-\lambda cNe^{r\lambda\tau^*_{PT}}e^{-(r+1)\lambda(\tau^*_{PT}+d)}\\&\Rightarrow\ \lambda cNe^{-\lambda\tilde\tau^*_{PT}}e^{-(r+1)\lambda\tilde d}-\lambda cNe^{-\lambda\tau^*_{PT}}e^{-(r+1)\lambda d}=\tilde k-k.\end{aligned}\tag{66}$$

(65) and (66) lead to

$$e^{-\lambda\tilde\tau^*_{PT}}e^{-(r+1)\lambda\tilde d}-e^{-\lambda\tau^*_{PT}}e^{-(r+1)\lambda d}\ge e^{-\lambda\tilde\tau^*_{PT}}-e^{-\lambda\tau^*_{PT}}.\tag{67}$$

All four terms in (67) are positive, so we must have

$$\begin{aligned}&e^{-(r+1)\lambda\tilde d}\ge e^{-(r+1)\lambda d}\ \Rightarrow\ \tilde d\le d\quad\text{or}\\&(\tilde T^*_{PT}-\tilde\tau^*_{PT})\le(T^*_{PT}-\tau^*_{PT}).\end{aligned}\tag{68}$$

Further examining (64) through (68), we conclude that $(\tilde T^*_{PT}-\tilde\tau^*_{PT})=(T^*_{PT}-\tau^*_{PT})$ holds if $m''(\tau)=0$ and $(\tilde T^*_{PT}-\tilde\tau^*_{PT})<(T^*_{PT}-\tau^*_{PT})$ holds if $m''(\tau)>0$.

Because $\tilde\tau^*_{PT}<\tau^*_{PT}$, we conclude from (68) that $\tilde T^*_{PT}<T^*_{PT}$.

(iii) **The impact of r.** Assume r increases to $\tilde{r} > r$. We denote the optimal solutions corresponding to $\tilde{r}$ by $\tilde{\tau}^*_{\mathrm{PT}}$ and $\tilde{T}^*_{\mathrm{PT}}$. Following a similar argument shown in (53) through (56), we conclude that $\tilde{\tau}^*_{\mathrm{PT}} > \tau^*_{\mathrm{PT}}$ holds with $\tilde{r} > r$. Further, from (8), we conclude

$$e^{\tilde{r}\lambda\tilde{\tau}^*_{\mathrm{PT}}} e^{-(\tilde{r}+1)\lambda\tilde{T}^*_{\mathrm{PT}}} = e^{r\lambda\tau^*_{\mathrm{PT}}} e^{-(r+1)\lambda T^*_{\mathrm{PT}}}. \tag{69}$$

Let $d = T^*_{\mathrm{PT}} - \tau^*_{\mathrm{PT}}$ and $\tilde{d} = \tilde{T}^*_{\mathrm{PT}} - \tilde{\tau}^*_{\mathrm{PT}}$. Then (69) can be rewritten as

$$e^{-\lambda\tilde{\tau}^*_{\mathrm{PT}}} e^{-(\tilde{r}+1)\lambda\tilde{d}} = e^{-\lambda\tau^*_{\mathrm{PT}}} e^{-(r+1)\lambda d}.$$

Because $e^{-\lambda\tilde{\tau}^*_{\mathrm{PT}}} < e^{-\lambda\tau^*_{\mathrm{PT}}}$, we must have

$$e^{-(\tilde{r}+1)\lambda\tilde{d}} > e^{-(r+1)\lambda d} \;\Rightarrow\; (\tilde{r}+1)\tilde{d} < (r+1)d \;\Rightarrow\; \tilde{d} < d \quad \text{or}$$

$$(\tilde{T}^*_{\mathrm{PT}} - \tilde{\tau}^*_{\mathrm{PT}}) < (T^*_{\mathrm{PT}} - \tau^*_{\mathrm{PT}}).$$

(iv) **The impact of the rate of increase in market opportunity cost.** Assume we have a new market opportunity cost function $\tilde{m}(\tau)$ with $\tilde{m}'(\tau) > m'(\tau)\ \forall\,\tau$. We denote the corresponding solution by $\tilde{\tau}^*_{\mathrm{PT}}$ and $\tilde{T}^*_{\mathrm{PT}}$. Analogous to (53) through (56), we have $\tilde{\tau}^*_{\mathrm{PT}} < \tau^*_{\mathrm{PT}}$.

Further, from (8), we get

$$e^{r\lambda\tilde{\tau}^*_{\mathrm{PT}}} e^{-(r+1)\lambda\tilde{T}^*_{\mathrm{PT}}} = e^{r\lambda\tau^*_{\mathrm{PT}}} e^{-(r+1)\lambda T^*_{\mathrm{PT}}}. \tag{70}$$

Because $e^{r\lambda\tilde{\tau}^*_{\mathrm{PT}}} < e^{r\lambda\tau^*_{\mathrm{PT}}}$, we must have

$$e^{-(r+1)\lambda\tilde{T}^*_{\mathrm{PT}}} > e^{-(r+1)\lambda T^*_{\mathrm{PT}}} \;\Leftrightarrow\; \tilde{T}^*_{\mathrm{PT}} < T^*_{\mathrm{PT}}.$$

Let $d = T^*_{\mathrm{PT}} - \tau^*_{\mathrm{PT}}$ and $\tilde{d} = \tilde{T}^*_{\mathrm{PT}} - \tilde{\tau}^*_{\mathrm{PT}}$. Then (70) can be rewritten as

$$e^{-\lambda\tilde{\tau}^*_{\mathrm{PT}}} e^{-(r+1)\lambda\tilde{d}} = e^{-\lambda\tau^*_{\mathrm{PT}}} e^{-(r+1)\lambda d}.$$

From $e^{-\lambda\tilde{\tau}^*_{\mathrm{PT}}} > e^{-\lambda\tau^*_{\mathrm{PT}}}$, we conclude

$$e^{-(r+1)\lambda\tilde{d}} < e^{-(r+1)\lambda d} \;\Rightarrow\; \tilde{d} > d \quad \text{or}$$

$$(\tilde{T}^*_{\mathrm{PT}} - \tilde{\tau}^*_{\mathrm{PT}}) > (T^*_{\mathrm{PT}} - \tau^*_{\mathrm{PT}}). \quad \square$$

A.8. Proof of Proposition 8

Even testing resource reallocation and uncertain market opportunity cost, the optimal testing stop time under the PT policy is always obtained by minimizing $E_{M(t)}[C_{\mathrm{PT}}(\tau, T)]$ as specified in (19), regardless of whether the release time is optimally determined or not. Assuming that the release time τ is given, we next derive the optimal testing stop time (T'^*_{PT}) that leads to the minimum expected cost. The first- and second-order derivatives with respect to T are

$$dE_{M(t)}[C_{\mathrm{PT}}(\tau, T)]/dT = k' - c\lambda' N e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\lambda')(T-R)}, \tag{71}$$

$$d^2E_{M(t)}[C_{\mathrm{PT}}(\tau, T)]/dT^2 = c\lambda'(r\lambda+\lambda') N e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\lambda')(T-R)} > 0. \tag{72}$$

From (71) and (72), we conclude that the optimal T'^*_{PT} satisfies

$$k' - c\lambda' N e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\lambda')(T'^*_{\mathrm{PT}}-R)} = 0$$

$$\Rightarrow\; N e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\lambda')(T'^*_{\mathrm{PT}}-R)} = k'/c\lambda'. \tag{73}$$

Note that (71) through (73) remain valid for any τ before time R. $N e^{-\lambda\tau} e^{-(r+1)\lambda(R-\tau)} e^{-(r\lambda+\lambda')(T'^*_{\mathrm{PT}}-R)}$ is the expected number of undetected bugs at the optimal testing stop time. Thus the proof is complete. $\square$

References

Arora, A., J. P. Caulkins, R. Telang. 2006. Research note—Sell first, fix later: Impact of patching on software quality. *Management Sci.* **52**(3) 465–471.

Baskerville, R., L. Levine, J. Pries-Heje, B. Ramesh, S. Slaughter. 2001. How internet software companies negotiate quality. *IEEE Comput.* **14**(4) 51–57.

Bass, F. M. 1969. A new product growth model for consumer durables. *Management Sci.* **15**(4) 215–227.

Berger, J. O. 1993. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York.

Chiu, K., J. Ho, Y. Huang. 2009. Bayesian updating of optimal release time for software systems. *Software Quality J.* **17**(1) 99–120.

Dalal, S. R., C. L. Mallows. 1988. When should one stop testing software? *J. Amer. Statist. Assoc.* **83** 872–879.

Dalal, S. R., C. L. Mallows. 1990. Some graphical aids for deciding when to stop testing software. *IEEE J. Selected Areas Comm.* **8**(2) 169–175.

Ehrlich, W., B. Prasanna, J. Stampfel, J. Wu. 1993. Determining the cost of a stop-test decision. *IEEE Software* **10**(2) 33–42.

Goel, A. L. 1985. Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. Software Engrg.* **SE-11**(12) 1411–1142.

Goel, A. L., K. Okumoto. 1979. Time-dependent error-detection rate model for software and other performance measures. *IEEE Trans. Reliability* **R-28**(3) 206–211.

Gross, N., M. Stepanek, O. Port, J. Carey. 1999. Software hell: Glitches cost billions of dollars and jeopardize human lives. How can we kill the bugs? *Bussiness Week* (December 6) 104–118.

McDaid, K., S. P. Wilson. 2001. Deciding how long to test software. *Statistician* **50**(2) 117–134.

Ohba, M., X. M. Chou. 1989. Does imperfect debugging affect software reliability growth? *Proc. 11th Internat. Conf. Software Engrg.*, IEEE Computer Society Press, Washington, DC.

Okumoto, K., A. L. Goel. 1980. Optimum release time for software systems based on reliability and cost criteria. *J. Systems Software* **1** 315–318.

Pham, H. 2000. *Software Reliability*. Springer, Singapore.

Pham, H. 2006. *System Software Reliability*. Springer, London.

Pham, H., X. Zhang. 1999. Software release policies with gain in reliability justifying costs. *Ann. Software Engrg.* **8**(1–4) 147–166.

Rinsaka, K., T. Dohi. 2006. Optimal testing and maintenance design in a software development project. *Electronics Comm. Japan, Part 3* **89**(6) 953–961.

Singpurwalla, N. D. 1991. Determining an optimal time interval for testing and debugging software. *IEEE Trans. Software Engrg.* **17**(4) 313–319.

Singpurwalla, N., S. Wilson. 1994. Software reliability modeling. *Internat. Statist. Rev.* **62**(3) 289–317.

Standish Group International, Inc. 2001. Extreme chaos. Retrieved July 13, 2011, http://standishgroup.com/sample_research/extreme_chaos.pdf.

Thibodeau, P. 2002. Study: Buggy software costs users, vendors nearly $60B annually. *Computerworld* (June 25). Retrieved July 13, 2011, http://www.computerworld.com/s/article/72245/Study_Buggy_software_costs_users_vendors_nearly_60B_annually.

Vienneau, R. L. 1991. The cost of testing software. *Reliability Maintainability Sympos.* (Jan. 29–31) 423–427.

Wood, A. 1996. Predicting software reliability. *IEEE Comput.* **29**(9) 69–77.

Xie, M., B. Yang. 2003. A Study of the effect of imperfect debugging on software development cost. *IEEE Trans. Software Engrg.* **29**(4) 471–473.