

# Optimal Software Release Time with Patching Considered

Zhengrui Jiang, Sumit Sarkar

School of Management, The University of Texas at Dallas

zxj011000@utdallas.edu, sumit@utdallas.edu

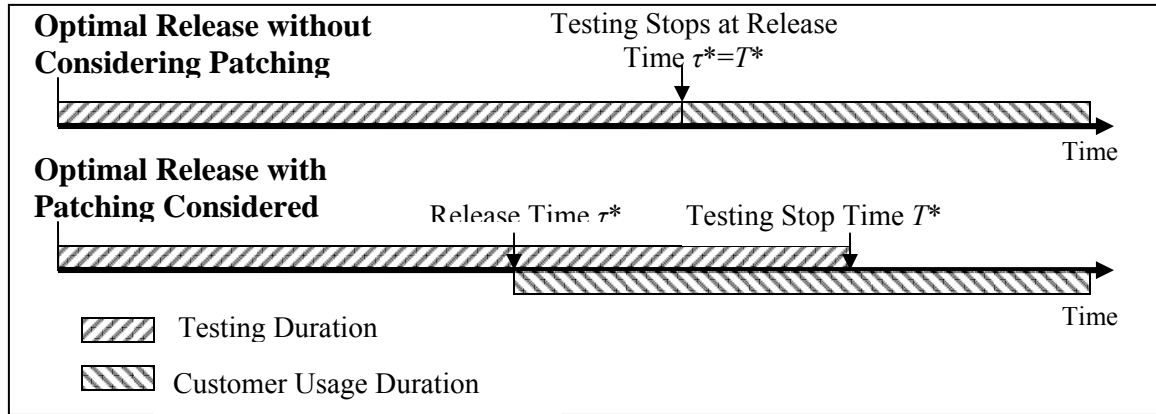
**Abstract:** Optimal software release policy is a well studied topic in the software reliability literature. However, all existing studies implicitly assume that testing stops at the time of release. Due to the advances in data communication technologies, an increasing number of software producers are providing patches for software defect correction and updates subsequent to a version release. Based on this fact, we propose an alternative release time scheduling policy – a software product is released before the optimal release time calculated by the traditional models and testing continues even after the software is released. Numerical solutions show that the optimal release policy with patching considered reduces the total cost by a significant amount.

## 1. Introduction

For any complex software project, one of the most important decisions project managers have to make is when to stop testing and ship the product to clients or to the market. If testing stops too early, many faults may remain uncovered. The software producer then risks dealing with dissatisfied customers and incurs a high cost of fixing bugs in the field. However, shipping a software product too late can be even more costly. Both empirical and theoretical studies have shown that there is a diminishing return to continued software testing effort. Besides the cost of testing, contract penalty and loss of market initiative may constitute an even bigger portion of the cost of late delivery. Therefore, both technical factors and economic factors have to be considered in deciding the optimal release time of a software product.

Since the software release time is such a crucial decision to make, the topic has been extensively studied in the past two decades and a number of optimal release policies have been proposed (Dalal and Mallows, 1988; Ehrlich et. al, 1993; Pham and Zhang, 1999; McDaid and Wilson, 2001). This stream of research is closely related to the broader software reliability literature, a good summary of which is provided by Pham (2000). In all these studies, the optimal release time is determined from the cost-benefit viewpoint. Testing should continue until the gain from the improved reliability cannot justify the cost of continued testing. An implicit assumption made in these studies is that software testing stops completely after release. After delivery, the task of “continued testing” is shifted to software users. Bugs are fixed after reported by users.

The development of commercial software systems is rarely a one-shot game. It is a common practice for software firms to continue to improve the quality of their software even after release. For many software products, improvements are realized by patching or updates. Advances in data communication technologies have significantly reduced the cost of software distribution and hence contributed to the frequent provision of software patching and updates. Given these developments, we believe that a software product can possibly be delivered earlier than the optimal release time calculated by the traditional models. Testing can continue even after the software product is released. Therefore, there will be a period of time during which the testers and customers are testing the software at the same time. Figure 1 illustrates the difference between the two practices. The goal of this study is to derive the optimal software release time and testing stop time when patching is considered, and compare the result with the case where patching is not considered.



**Figure 1. Optimal Release Time with and without Patching considered**

## 2. The Reliability Assumptions

We discuss four software reliability assumptions in this section. The first three assumptions are also the underlying assumptions for a number of commonly used software reliability models such as the Jelinski-Moranda model (Jelinski and Moranda, 1972), which is “credited with being the first model for assessing software reliability” (Gokhale et al, 1996), and the Goel-Okumoto Non-homogenous Poisson Process model (Goel and Okumoto, 1979), which is frequently adopted in software release policy studies (Pham and Zhang, 1999; McDaid and Wilson, 2001).

**Assumption 1.** Each bug in a software system is independent and equally likely to cause a failure during a test, and the amount of additional testing time it takes to find an undetected bug is independent of the amount of time testers have already spent. This implies that the lifetime of all bugs follows an iid exponential distribution. We denote the failure rate of all bugs by  $\lambda$ . The density of lifetime distribution of all bugs is given by:

$$f(t) = \lambda \cdot e^{-\lambda t} . \quad (1)$$

Thus, the cumulative distribution, which is the probability that a bug will be detected by time  $t$ , is given by:

$$F(t) = 1 - e^{-\lambda t} . \quad (2)$$

**Assumption 2.** The total number of bugs in a software system just before testing starts is a fixed finite number  $N$ .

**Assumption 3.** Once a bug is found by testers or customers, it is fixed instantaneously and without causing any additional errors.

**Assumption 4.** Costumers’ usage of a software system is similar to testers’ testing in that all bugs are independent and equally likely to be detected by a customer and the remaining lifetime of a bug is independent of how long the software system has been in use. However, the rates of bug detection by testers and customers are not expected to be equal. The differences in the rate of detection, reflected by different values of  $\lambda$ , arise from different *intensities* and *efficiencies* of testing by the two groups, and the different sizes of the two groups. *Intensity* of testing (usage) refers to the average amount of time spent in testing (usage) in one day. Testers usually spend all their working hours on testing, or even have the software tested 24 hours a day by using some automated testing tools. Customers, on the other hand, may use the software only two hours a day. Therefore, their intensities of testing (usage) are different. *Efficiency* of testing refers to how effective time is spent in detecting errors. When the same amount of time is spent on testing and on usage, we expect that a bug is more likely to be detected by a tester than by a customer. This

is due to the better availability of testing tools and professional training to testers. We denote the failure rate of a bug under customer usage by  $\lambda' = r\lambda$ , where  $r$  is the ratio of fault detection rate under customers' usage with respect to testers' testing. If the customer population is homogeneous, we can claim that  $r$  is proportional to the product of the intensity and efficiency of "testing" by customers and the size of the customer base.

### 3. The Cost Model

Since the goal of this paper is to determine the optimal release time, we only need to consider the costs that are affected by the amount of testing time. All other cost or benefit factors that do not depend on the release time or testing stop time are deemed constant and therefore not included in our model. We identify three major cost factors that can be affected by the release time, namely, the *cost of testing*, the *cost of software failure in the field*, and the *market opportunity cost*. The *cost of testing* refers to the cost of testers' activities such as test planning, test case generation, test execution, and test result analyzing. We adopt the assumption by Ehrlich et. al. (1993) and Pham and Zhang (1999) that the cost of testing is a linear function of the amount of testing time. The *cost of software failure in the field* is incurred when a failure is reported by one of the customers. We also assume that once a failure is reported by a customer, it is removed instantaneously and patches are sent to all other customers such that the same bug will not cause any more problems. The cost of software failure in the field includes both the direct cost associated with bug removal in the field and the indirect cost such as the liability cost and the loss of goodwill resulting from customers' dissatisfaction. We assume the total cost of software failures in the field is proportional to the number of defects not discovered by testers. Similar assumptions are made by Dalal and Mallows (1988) and Ehrlich et al. (1993). The *market opportunity cost* refers to the direct and indirect costs related to the market or contract. Examples include the contract penalties if the scheduled delivery time is missed, and the loss of market initiatives to competitors. We assume that the market opportunity cost is a monotonically increasing convex function of time. The *cost of fixing a bug found during testing* is formulated as a separate cost in some studies (Dalal and Mallows, 1988; Ehrlich et al., 1993; Pham and Zhang, 1999). Since this cost is significantly smaller than the cost of failure caused by a bug in the field, we assume it to be negligible in our model. We also assume that *the cost of patching* to fix a bug in a delivered software product is negligible, no matter the bug is detected by testers or one of the customers.

<p><math>N</math> – Total number of bugs in the software when testing begins  <math>T</math> – Testing stop time  <math>\tau</math> – Release time  <math>u(\tau)</math> – Expected number of undetected bugs at the time of release  <math>k</math> – Cost of testing per unit time  <math>a</math> – Cost of one software failure in the field</p>
--

**Figure 2. Cost Model Parameters**

The optimal release policies can be obtained by minimizing the total of the three major costs. In what follows, we first show the optimal release time without considering patching and then derive the optimal release time and the optimal testing stop time when patching is taken into consideration. The parameters shown in Figure 2 are used in the derivation of the three cost factors. The expected number of undetected bugs at the time of release is given by:

$$u(\tau) = N \cdot e^{-\lambda\tau}. \quad (3)$$

#### 4. Optimal Release Policy without Considering Patching

When patching is not considered, testing stops at the time of release, which implies  $\tau = T$ . We now derive the different cost factors for this case.

- (1) Based on the assumption that the cost of testing, denoted by  $C_1(\tau)$ , is linear with the amount of testing time, we have

$$C_1(\tau) = k \cdot \tau. \quad (4)$$

- (2) The expected cost of software failure in the field, denoted by  $C_2(\tau)$ , is assumed to be linear with the expected number of undetected bugs at the time of release, i.e.,

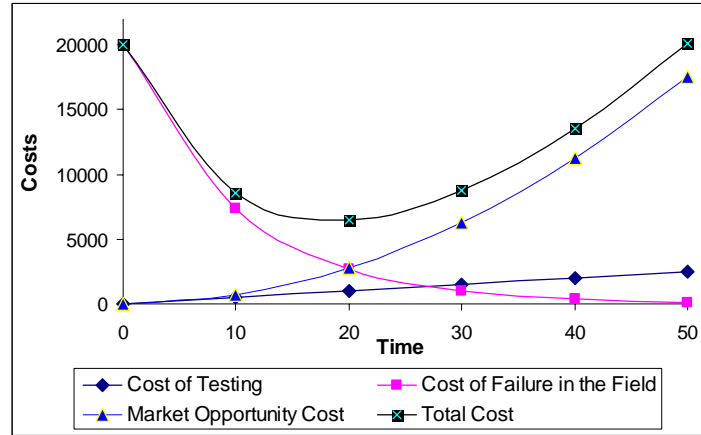
$$C_2(\tau) = a \cdot u(\tau) = a \cdot N \cdot e^{-\lambda\tau}. \quad (5)$$

- (3) The market opportunity cost, denoted by  $C_3(\tau)$ , is assumed to be a monotonically increasing, twice continuously differentiable convex function of  $\tau$ . Since the qualitative conclusion of the study is not much affected by the actual functional form of  $C_3(\tau)$ , we use the simple quadratic function as in (6):

$$C_3(\tau) = m \cdot \tau^2, \quad (m > 0). \quad (6)$$

- (4) Summing over the three cost factors, we get the expected total cost:

$$EC(\tau) = k \cdot \tau + a \cdot N \cdot e^{-\lambda\tau} + m \cdot \tau^2. \quad (7)$$



**Figure 3. Cost Model**

A simple illustration of the various costs is given in Figure 3. The optimal release time  $\tau^*$  without patching can be obtained by minimizing (7) with respect to  $\tau$ . We assume the optimal release time is an interior solution. Since  $EC(\tau)$  is convex in  $\tau$ , we can use the first order condition to obtain the optimal release time. The following solution is obtained by the mathematical software MATHEMATICA.

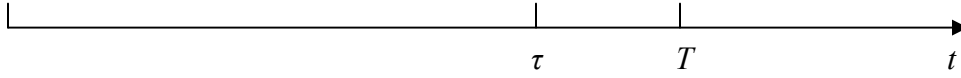
$$\tau^* = \frac{-k\lambda + 2m \cdot \text{ProductLog}[a \cdot e^{\frac{k\lambda}{2m}} \cdot N \cdot \lambda^2 / 2m]}{2m\lambda}. \quad (8)$$

The optimal expected cost is given by:

$$EC^* = EC(\tau^*) = k \cdot \tau^* + a \cdot N \cdot e^{-\lambda\tau^*} + m \cdot (\tau^*)^2. \quad (9)$$

## 5. Optimal Release Time with Patching Considered

As shown in Figure 1, when patching is considered, a software system can be released before testing stops. The complete solution with patching considered involves the determination of the two optimal times, i.e., the optimal release time and the optimal testing stop time. As illustrated by Figure 4, in this case the whole time horizon is divided into three parts. Before the total cost can be derived, we first examine the probability that a bug is detected in each of the three different periods.



**Figure 4. Time Horizon Divided by  $\tau$  and  $T$**

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

$$F_1(\tau, T) = 1 - e^{-\lambda\tau} . \quad (10)$$

- (2) 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 product of the probability that the bug is detected in  $(T-\tau)$  amount of time, and the probability that it is not found in the first period, which gives (11).

$$F_2(\tau, T) = e^{-\lambda\tau} \cdot (1 - e^{-(r+1)\lambda(T-\tau)}) = e^{-\lambda\tau} - e^{r\lambda\tau} \cdot e^{-(r+1)\lambda T} . \quad (11)$$

If a bug is detected in the second period, the conditional probability that it is detected by one of the customers is:

$$P(\text{CustomerDetection} \mid \text{Detected\_In\_Period2}) = \frac{r \cdot \lambda}{r \cdot \lambda + \lambda} = \frac{r}{r+1} . \quad (12)$$

- (3) The probability that a bug is not detected until the last period is given by:

$$F_3(\tau, T) = e^{-\lambda\tau} \cdot e^{-(r+1)\lambda(T-\tau)} = e^{r\lambda\tau} \cdot e^{-(r+1)\lambda T} . \quad (13)$$

We now examine the total expected cost for this case. Based on the probabilities above, we get the total expected cost when patching is considered:

$$\begin{aligned} EC_P(\tau, T) &= k \cdot T + a \cdot N \left[ F_2(\tau, T) \frac{r}{r+1} + F_3(\tau, T) \right] + m \cdot \tau^2 \\ &= k \cdot T + a \cdot N \frac{r}{r+1} e^{-\lambda\tau} + \frac{a \cdot N}{r+1} e^{r\lambda\tau} \cdot e^{-(r+1)\lambda T} + m \cdot \tau^2 . \end{aligned} \quad (14)$$

It can be verified that the Hessian of  $EC_P$  at any point is positive definite. Therefore the optimal release time  $\tau_p^*$  and testing stop time  $T_p^*$  can be calculated from the first order conditions. A closed form expression could not be derived. However, since the local minimum is also the global minimum, a numerical solution can be quickly obtained using a search method.

## 6. Numerical Study

To compare the two policies, namely, the optimal release time scheduling with and without considering patching, we conducted the following numerical study. Initially, the parameters used were  $k = 50$ ,  $\lambda = 0.1$ ,  $N = 1000$ ,  $a = 20$ ,  $m = 7$ , and  $r = 0.4$ . The results are as follows. (1) Without patching:  $\tau^* = T^* = 19$ ,  $EC^* = 6468$ . (2) With patching:  $\tau_p^* = 12$ ,  $T_p^* = 30$ ,  $EC_p^* = 4575$ . As expected, when patching is considered, the optimal release time  $\tau_p^*$  is earlier than  $\tau^*$ , the optimal release time obtained without considering patching, and the optimal testing stop time  $T_p^*$  is later

than  $T^*$ , the optimal stop time without considering patching. With patching considered, we achieve a cost saving of 30%, which is a significant reduction in cost.

By varying the values of all six parameters, we repeated the study for 25 times. In all cases, the inequalities  $\tau_p^* < \tau^* = T^* < T_p^*$  and  $EC_p^* < EC^*$  hold. The average cost saving in these cases is 25%. This shows that the optimal release policy we propose here is superior to the existing solutions without considering patching.

## 7. Conclusion and Discussion

Due to the advances of data communication technologies, software producers can provide software updates and bug fixes using patches at a very low cost. Therefore, in determining the optimal release time, software producers can choose to release product earlier than the time suggested by the traditional approach, and continue to test the software after it is released. After release, once a bug is discovered by testers or users, a patch is issued. Based on this modified software release policy, a significant amount of cost, which is predominantly the market opportunity cost, can be saved.

We used some strong assumptions in this paper. We assume that all bugs are iid distributed. The identical part of the assumption can be relaxed without changing the qualitative result of the model. We also assume that the cost of fixing bugs found during testing and the cost of patching to be negligible in this preliminary study. If we assume these two costs to be linear with the number of bugs detected and significantly smaller than the cost of software failure in the field and the market opportunity cost, the conclusions of this study will still be valid.

In this study, the optimal release time and testing stop time are determined at the same time based on known parameters. An extension of this study would be to develop a two-stage or multi-stage decision model to decide first the release time condition and then testing stop condition based on the observed failure intervals up to the decision time.

## References

- Dalal, S.R., and C.L. Mallows. 1988. When should one stop testing software? *Journal of the American Statistical Association*, vol. 83, pp. 872-879.
- Ehrlich, W., B. Prasanna, J. Statmpfel, and J. Wu. 1993. Determining the cost of a stop-test decision. *IEEE Software*, March 1993.
- Goel, A.L. and K. Okumoto. 1979. Time-dependent error-detection rate model for software and other performance measures, *IEEE Transactions on Reliability*, vol. R-28 (3), 1979.
- Gokhale, S.S., P.N. Marinos, and K.S. Trivedi. 1996. Important milestones in software reliability modeling, *Proc. Of Software Engineering and Knowledge Engineering*, Lake Tahoe, NV.
- Jelinski, Z. and P.B. Moranda. 1972. Software reliability research, in *Statistical Computer Performance Evaluation*. W. Freiberger (ed.), Academic Press, New York.
- Pham, H. 2000. *Software Reliability*, Springer, Singapore.
- Pham, H, and X. Zhang. 1999. Software release policies with gain in reliability justifying costs. *Annals of Software Engineering*, vol. 8, 147-166.
- McDaid, K. and S.P. Wilson. 2001. Deciding how long to test software. *The Statistician*, vol. 50, Part 2, 117-134.