Dynamic release policies for software systems with a reliability constraint

SHAOHUI ZHENG

School of Business and Management, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
E-mail: imzheng@ust.hk

Received September 1999 and accepted December 2000

We discuss the optimal release problem of computer software. A conditional non-homogeneous Poisson process model is used to describe the software reliability growth behavior. By formulating with Markov decision programming, we show that, to minimize the total discounted cost, the optimal release policy is threshold-type, which is easy to obtain and to implement. It is then extended to the model with a constraint on the system reliability, for which a similar threshold-type control policy is proved to be optimal too.

1. Introduction

As computer software becomes more and more sophisticated, whilst becoming more and more important in our lives, people are increasingly concerned about its reliability. Many different models, generally called Software Reliability Growth Models (SRGMs) have been developed to describe the failure patterns of software systems. One of these models, developed by Goel and Okumoto (1979), is the Non-Homogeneous Poisson Process (NHPP) model, which assumes that \( N(t) \), the number of faults (bugs) detected up to time \( t \), follows a NHPP. Another model, introduced by Jelinski and Moranda (1972), is the de-eutrophication model. It assumes that the total number of faults at the start of testing is a known constant \( N \), and the system failure rate at any time is proportional to the fault content of the program at that moment. A good survey on different software reliability growth models can be found in Goel (1985) or Lyu (1996).

Generally, the effort involved in improving the reliability of a complex software system consists of a very time-consuming and expensive testing/debugging procedure. It is reported that in many cases, more than half of the time and cost is spent in system testing when developing a computer software. Hence, one of the important problems in this effort is when we should stop testing and are ready to release the system for use.

Obviously, if testing stops too soon, then there will be too many faults left in the program, which will result in excessive system failures during operation, and lead to significant losses because of the failure penalty or customer dissatisfaction. On the other hand, spending too much effort in testing may result in an unnecessarily high testing cost, and delay the introduction of the product into the market place.

In the literature, many different approaches have been developed to determine the optimal release time of software systems, based on different SRGM. See, for instance, Okumoto and Goel (1980); Koch and Kubat (1983); Shanthikumar and Tufekci (1983); Yamada and Osaki (1987); Dalal and Mallows (1988); Ohtera and Yamada (1990); Leung (1992); Xie and Hong (1998). In most of these papers, it is assumed that the parameters in the related SRGM are known constants, and the problems are modeled as a static optimization problem, by solving until an optimal release time \( T^{*} \) is obtained. The software will then be released after testing \( T^{*} \) units of time, no matter what outcome is observed during testing. Dynamic approaches are discussed in Shanthikumar and Tufekci (1983) and Dalal and Mallows (1988). In Shanthikumar and Tufekci (1983), the de-eutrophication model of Jelinski and Moranda (1972) is used as the reliability model. As the initial number of faults in the system is assumed to be a known constant, it is then argued that, if more faults are detected, less faults will remain in the system. Hence, it is optimal to stop testing if the number of faults detected is greater than a predetermined value. These model settings seem to be unpleasant and impractical in some cases, since: (I) it is generally not easy, if not impossible, to have an accurate estimation of the model parameters (such as the number of faults in the system) before intensive testing; and, (II) using the same testing effort and testing approach, if more faults are detected, then the system reliability is likely to be lower.
and more testing may be required. A more attractive dynamic model is studied in Dalal and Mallows (1988). Extending the de-eutrophication model, it is assumed in Dalal and Mallows (1988) that the total number of faults \( N \), is a Poisson-distributed random variable with parameter \( \lambda \), and \( \lambda \) is another random variable with a Gamma distribution. A dynamic control rule is then proved to be optimal for this problem.

In this paper, we model the SRGM with a conditional NHPP, of which the mean value function depends on a random variable \( X \). Here \( X \) represents the general quality of the computer program and may have any distribution. A sequential approach is used to update our knowledge on \( X \) during testing, which results in a dynamic procedure for deciding the system release time. We show that, to minimize the total discounted cost, the optimal control policy has the following threshold-type structure: at time \( t \), testing should be stopped if and only if \( n < n_t \), where \( n \) is the number of faults detected, and \( n_t \) is a predetermined threshold value. Clearly, this policy is opposite to that in Shanthikumar and Tufekci (1983). We then extend the model to consider the problem with a reliability constraint. A similar threshold-type policy is proved to be optimal for the constrained problem. The relationships of optimal policies for constrained and non-constrained problems are also discussed. The problems are formulated with Markov Decision Programming ((MDP) or stochastic dynamic program) in this paper. (For constrained MDP problems, readers can also refer to Feinberg (1994) and Yao and Zheng (1999).)

Comparing to Dalal and Mallows (1988), our model setting is more general, as \( X \) is a random variable with any distribution. In particular, as mentioned in Section 6, Dalal and Mallows (1988) is actually just one of our special cases for the non-constrained model. Besides, since the reliability constraint is considered, the problems discussed here are more applicable.

The rest of this paper is organized as follows. The model is described in detail, and a MDP formulation is given in Section 2. In Section 3, monotonicities of some system characteristics are derived. We prove the threshold property of the optimal policy for the non-constrained problem in Section 4, and generalize it to the model with a reliability constraint in Section 5. The paper finishes with conclusions in Section 6, where some extensions to this work are also discussed.

### 2. Model description and dynamic programming formulation

Suppose each fault in the software may cause a system failure during testing or operating periods. When the system fails, the related fault will be detected and removed immediately. In this paper, we assume that debugging is perfect and takes no time, i.e., a fault will be definitely removed when detected, and no new fault will be introduced into the system during the debugging. Let \( C_i \) denote the testing cost per unit of time. If a fault is detected during testing, it will be removed with a cost \( C_i \). Otherwise, if a fault causes a failure (and it will then be detected and removed) during operation after release, the fixing and penalty cost is \( C_R \), with \( C_R \gg C_i \). Suppose the release deadline is \( T_D \), i.e., the system should be released at or before time \( T_D \). At time \( T_D \), if the software reliability is shown to be too low so that it is impossible or too risky to release the system, then we can choose to crash the system and not to deliver it by paying a default penalty \( C_P \). Hence, there are two decision alternatives for each time period: to release or to crash the system for \( t = T_D \), and to release or to continue testing the system for \( t < T_D \). If the system is released, its life cycle (or warranty period) is supposed to be \( T_L \), in which the fixing and penalty costs of operation failures should be considered.

Suppose \( X \) be a random variable that represents the average initial fault content in the system. It is assumed that \( X \) is known only through its distribution, and when \( X = x \), \( N(t) \), the number of faults detected up to time \( t \), follows a NHPP with mean function and intensity function

\[
m_X(t) = x(1 - e^{-\mu t}) \quad \text{and} \quad \lambda_X(t) = m_X(t) = x \mu e^{-\mu t},
\]

respectively. Here \( \mu \) is a known constant representing the fault detection rate per fault. A random variable \( X \) reflects the great variability of the software quality due to variations of the system complexity and the proficiency levels of the programmers etc. In this paper, we have placed no restriction on the distribution function of \( X \).

**Remark.** Suppose at the beginning of testing, \( N \), the total number of faults in the system, is a Poisson-distributed random variable with parameter \( X \), while \( X \) itself is also a random variable. (Hence, \( N \) is actually conditionally Poisson distributed.) Also suppose that faults are independent and the time to detect a fault is exponentially distributed with mean \( \mu \). Then we can easily show that \( N(t) \), the number of faults detected up to time \( t \), will follow the conditional NHPP described above.

For convenience, let \( X_n(t) \) denote the conditional realization of \( X \) for given \( N(t) = n \). That is,

\[
P[X_n(t) = x] := P[X = x | N(t) = n],
\]

for \( x, n \geq 0 \) and \( t \geq 0 \). Let \( D_n(t) \) denote the number of faults detected within a unit of time following \( t \), given \( N(t) = n \), i.e.,

\[
P[D_n(t) = d] := P[N(t + 1) - N(t) = d | N(t) = n]
\]

for \( d = 0, 1, \ldots \).

To decide the optimal release time of the system, we formulate the problem with discrete time Markov decision programming as follows.