Optimal Stopping in Software Testing

Nilgun Morali,[1] Refik Soyer[2]

[1] *Department of Statistics, Dokuz Eylal Universitesi, Turkey*

[2] *Department of Management Science, The George Washington University, 2115 G Street, NW, Washington, DC 20052*

Received 30 November 1998; revised 18 June 2002; accepted 28 June 2002

DOI 10.1002/nav.10048

Abstract: In this paper we address the problem of how to decide when to terminate the testing/modification process and to release the software during the development phase. We present a Bayesian decision theoretic approach by formulating the optimal release problem as a sequential decision problem. By using a non-Gaussian Kalman filter type model, proposed by Chen and Singpurwalla (1994), to track software reliability, we are able to obtain tractable expressions for inference and determine a one-stage look ahead stopping rule under reasonable conditions and a class of loss functions. © 2002 Wiley Periodicals, Inc. Naval Research Logistics 50: 88–104, 2003.

Keywords: Bayesian inference; stochastic ordering; dynamic programming; Kalman filter; preposterior analysis; software reliability

1. INTRODUCTION AND OVERVIEW

During the development phase, software is subjected to several stages of testing to identify existing problems. At the end of each test stage, corrections and modifications are made to the software with the hope of increasing its reliability. This process is called *reliability growth.* It is possible, however, that a particular modification, or a series of modifications, could lead to a deterioration in performance of the software. The two key statistical issues are:

(i) how to model and how to describe the changes in the performance of the software as a result of the modifications,

(ii) how to decide when to terminate the testing/modification process and to release the software.

Most of the literature in software reliability centers around the first issue, namely, that of modeling and inference for assessment of software reliability and neglects the important issue

Correspondence to: R. Soyer

of determining when to stop the testing process and release the software. Exceptions to these are the works Forman and Singpurwalla [5, 6], Okumoto and Goel [13], Yamada, Narihisa, and Osaki [19], and Ross [15]. None these of approaches are based on a formal decision-theoretic approach.

Dalal and Mallows [3] and Singpurwalla [16, 17] propose a Bayesian decision theoretic approach to the problem. The work of Dalal and Mallows [3] provides an exact but a complicated solution, and, as a result, an asymptotic solution is given by the authors. The work of Singpurwalla [16, 17] addresses a two-stage problem where the solution is complicated due to computational difficulties involved in preposterior analysis. More recent work by Ozekici and Catkan [14] and McDaid and Wilson [11] also propose decision theoretic solutions. Ozekici and Catkan [14] present a very general setup and provide characterizations of the optimal release policy. The proposed approach does not include any revision of uncertainty (or learning) in the Bayesian sense and computational difficulties arise due to the recursive nature of the dynamic programming solution. The work of McDaid and Wilson [11] considers the case of single-stage testing using nonhomogeneous Poisson process type models and develops a Bayesian decision theoretic solution. The authors also consider a sequential testing problem and show that the solution is analytically intractable.

In this paper, we present a Bayesian decision theoretic approach by formulating the optimal release problem as a sequential decision problem. The solution of sequential decision problems of this type is usually too difficult to analyze due to the reliance on preposterior analysis (see, e.g., van Dorp, Mazzuchi, and Soyer [18]). Here, by using a non-Gaussian Kalman filter type of model suggested by Chen and Singpurwalla [2] to track software reliability, we are able to obtain tractable expressions for inference and determine a one-stage look ahead stopping rule under reasonable conditions and a class of loss functions.

One of the strengths of the Bayesian approach is that it allows for the incorporation of prior perceptions about software testing process into the analysis and provides a formalism for combination of this subjective input with available test data via the calculus of probability. In addition, the Bayesian approach provides a coherent framework for making decisions (see, e.g., Lindley [10]) with regards to optimal stopping during the software development phase. The Bayesian decision-theoretic formulation of the problem enables us to process information sequentially as we learn about failure behavior of the software and to adapt our optimal stopping strategy accordingly. In other words, the Bayesian approach is adaptive in that it allows for revision of our stopping strategies as well as our uncertainties in a dynamic manner. Thus, it is very suitable for sequential decision problems.

A synopsis of our paper is as follows: In Section 2, we formulate the optimal stopping problem as a sequential decision problem and present the conditions for optimality of one-stage look ahead stopping rule for a general loss function. In Section 3, we present a version of the non-Gaussian Kalman filter type of model suggested by Chen and Singpurwalla [2] for monitoring software reliability and discuss sequential inference and m-step ahead predictions for the model. In Section 4, we consider specific forms of loss function and using the model of Section 3 develop the conditions under which the one-step look ahead rule is optimal. Finally, in Section 5, we illustrate the use of the stopping rule with two examples.

2. THE SEQUENTIAL STOPPING PROBLEM

Let X_i, $i = 1, 2, \ldots$, denote the life-length of the software during the ith stage of testing after the $(i - 1)$st modification made to it. A common view in software reliability modeling is that, unlike a hardware component, software does not exhibit an aging or wearout behavior. It is assumed that the failure rate of the software is constant during the ith stage of testing and is denoted by θ_i. Thus, given θ_i the failure behavior of the software during the ith stage of testing is described by the exponential density

$$p(x_i \mid \theta_i) = \theta_i \exp(-\theta_i x_i). \tag{1}$$

The special feature of the model (1) is that it allows the failure rate θ_i to change from one testing stage to another as a result of corrections made to the software. At the end of each stage, following modifications made to the software, a decision must be made whether to terminate the debugging process. Thus after completion of i stages of testing, the decision of whether or not to stop testing will be based on $X^{(i)} = \{X^{(0)}, x_1, x_2, \ldots, x_i\}$, where $X^{(0)}$ represents available information concerning failure characteristics of software prior to any testing.

In the Bayesian decision-theoretic framework, the decision to whether or not stop testing must be based on maximization (minimization) of expected utility (loss). Any reasonable loss (or utility) function should consider the tradeoff between the loss associated with extensive testing versus the loss associated with releasing an unreliable piece of software. In consideration of such tradeoffs, we define the loss associated with stopping and releasing the software after the ith testing stage as

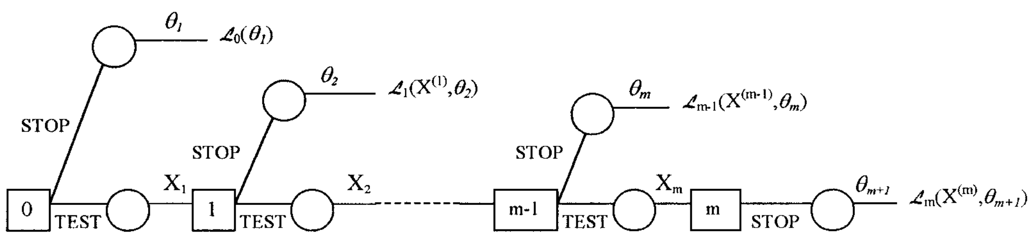
$$\mathscr{L}_i(X^{(i)}, \theta_{i+1}) = \sum_{j=1}^{i} \mathscr{L}_T(X_j) + \mathscr{L}_S(\theta_{i+1}), \tag{2}$$

where $\mathscr{L}_T(\cdot)$ represents the loss due to testing for one stage, and $\mathscr{L}_S(\cdot)$ relates the loss associated with stopping and releasing the software. We note that $\mathscr{L}_0$ (the loss associated releasing the software before any testing) is a function of θ_1 (the failure rate prior to any modification) since $\Sigma_{j=1}^{0} \{\cdot\} \equiv 0$.

We note that, in (2), $\mathscr{L}_T(\cdot)$ typically captures the cost of testing. $\mathscr{L}_T$ is a function of X_j since it is assumed that at each stage the software is tested until failure. This is a reasonable assumption at the development phase since there will be typically a large number of bugs present in any complex software. This is also a common assumption in reliability growth testing (see, e.g., Erkanli, Mazzuchi and Soyer [4]). The real software failure data which is used in Section 5.2 for illustration also fits into this type of testing scenario. $\mathscr{L}_T(\cdot)$ will be an increasing function of testing time X_j. In (2), $\mathscr{L}_S(\cdot)$, the loss associated with stopping and releasing the software, typically reflects costs associated with bugs discovered and fixed after testing, that is, it deals with failures during the usage phase. Since the failure rate θ_{i+1} of the software will be proportional to the number of bugs present in software after release, $\mathscr{L}_S(\cdot)$ will be assumed to be an increasing function of θ_{i+1}.

The stopping problem can be represented as a sequential decision problem as given by the m-stage decision tree in Figure 1 and can be solved using dynamic programming. Solution of the tree proceeds in the usual way by taking expectation at random nodes and minimizing the expected loss at the decision nodes. At decision node i, the *additional expected loss* associated with the STOP and the TEST decisions are given by the terms $E[\mathscr{L}_S(\theta_{i+1}) \mid X^{(i)}]$ and $E[\mathscr{L}_T(X_{i+1}) \mid X^{(i)}] + L^*_{i+1}$, respectively where

$$L^*_i = \text{MIN}\{E[\mathscr{L}_S(\theta_{i+1}) \mid X^{(i)}], E[\mathscr{L}_T(X_{i+1}) \mid X^{(i)}] + L^*_{i+1}\} \tag{3}$$



θ_1
$\mathscr{L}_0(\theta_1)$
STOP
0
TEST
X_1
θ_2
$\mathscr{L}_1(X^{(1)},\theta_2)$
STOP
1
TEST
X_2
θ_m
$\mathscr{L}_{m-1}(X^{(m-1)},\theta_m)$
STOP
m-1
TEST
X_m
m
STOP
θ_{m+1}
$\mathscr{L}_m(X^{(m)},\theta_{m+1})$

Figure 1. Decision tree for software release problem.

for $i = 0, 1, \ldots,$ and the optimal decision at decision node i then is the one associated with L_i^*.

In Figure 1, m, the maximum number of testing stages, can be considered infinite with $L_{m+1}^* = \infty$. We note that even for the case of finite m the calculation of L_i^* in (3) is not trivial as it involves implicit computation of expectations and minimizations at each stage. We can rewrite (3) as

$$L_i^* = \underset{\delta=0,1,\ldots}{\text{MIN}} \{L_i^{(\delta)}\}, \tag{4}$$

where

$$L_i^{(\delta)} = \sum_{j=1}^{\delta} \text{E}[\mathscr{L}_\text{T}(X_{i+j}) \mid X^{(i)}] + \text{E}[\mathscr{L}_\text{S}(\theta_{i+\delta+1}) \mid X^{(i)}] \tag{5}$$

is the additional expected loss associated with testing for δ more stages after the ith modification to the software.

Given the above development, the following theorem provides an optimal stopping rule.

THEOREM 1: Let $\text{E}[\mathscr{L}_\text{T}(X_j) \mid X^{(i)}]$ be increasing in j for $j = i + 1, \ldots,$ and $\text{E}[\mathscr{L}_\text{S}(\theta_j) \mid X^{(i)}]$ be discrete convex in j for $j = i + 1, \ldots,$ then after the ith modification to the software, the following stopping rule is optimal:

$$\text{if} \begin{cases} L_i^{(1)} - L_i^{(0)} < 0 \Rightarrow & \text{Continue Testing} \\ L_i^{(1)} - L_i^{(0)} \geq 0 \Rightarrow & \text{Stop Testing and Release} \end{cases} \tag{6}$$

PROOF: The proof follows along the same lines as in van Dorp, Mazzuchi, and Soyer [18]:
If $L_i^{(1)} - L_i^{(0)} < 0$, then continue testing is the optimal action.
If $L_i^{(1)} - L_i^{(0)} \geq 0$, then

$$\text{E}[\mathscr{L}_\text{T}(X_{i+1}) \mid X^{(i)}] \geq \text{E}[\mathscr{L}_\text{S}(\theta_{i+1}) \mid X^{(i)}] - \text{E}[\mathscr{L}_\text{S}(\theta_{i+2}) \mid X^{(i)}]..$$

Discrete convexity of $\text{E}[\mathscr{L}_\text{S}(\theta_j) \mid X^{(i)}]$ in j for $j \geq i + 1$ implies that the differences $\{\text{E}[\mathscr{L}_\text{S}(\theta_{j+1}) \mid X^{(i)}] - \text{E}[\mathscr{L}_\text{S}(\theta_j) \mid X^{(i)}]\}$ are increasing for $j \geq i + 1$. This together with the assumption that $\text{E}[\mathscr{L}_\text{T}(X_j) \mid X^{(i)}]$ is increasing in j for $j \geq i + 1$ guarantees that $L_i^{(\delta+1)} - L_i^{(\delta)} \geq 0$, for $\delta \geq 0$ and thus stop testing and release is the optimal decision. □

Thus, under the given conditions by the theorem, a one-stage look ahead rule is optimal. The theorem implies that it is optimal to stop testing when the expected increase in loss due to testing an additional stage is greater than the expected decrease in loss due to the improvement in failure rate resulting from testing an additional stage.

3. A MODEL FOR THE FAILURE RATE OF SOFTWARE

As pointed out in Section 2, the failure rate of the software is constant during the ith stage of testing and therefore given θ_i, the life-length of the software at the ith stage, is assumed to be exponentially distributed as in (1). Due to the modifications made to the software, it is expected that the failure rate at the ith stage will be related to the failure rate at the previous stages. Thus, we assume that given $X^{(i-1)}$ and θ_{i-1}

$$\left(\frac{\theta_i}{\rho\theta_{i-1}}\middle|\, \theta_{i-1}\right) \sim \mathrm{Beta}(\gamma\alpha_{i-1}, (1-\gamma)\alpha_{i-1}), \tag{7}$$

where ρ, γ, and α_{i-1} are known nonnegative quantities such that $0 < \gamma < 1$. We note that (7) can be written as

$$\theta_i = \rho\theta_{i-1}\xi_i \tag{8}$$

with $\xi_i \sim \mathrm{Beta}(\gamma\alpha_{i-1}, (1-\gamma)\alpha_{i-1})$. Equation (8) can be thought of as a state equation, and it provides an stochastic ordering of θ_i's for $\rho > 1$ since the above implies that $\theta_i < \rho\theta_{i-1}$ for all i. Furthermore, at stage 0, given $X^{(0)}$, we assume that θ_0 has a gamma distribution, with shape parameter α_0 and scale parameter β_0 denoted as

$$(\theta_0 \mid X^{(0)}) \sim \mathrm{Gamma}(\alpha_0, \beta_0). \tag{9}$$

The distributional assumptions given by (7) and (9) enable us to develop an analytically tractable Bayesian analysis. The resulting posterior and predictive distributions make the sequential inference and decision making feasible in our problem. In addition, we would like to note that these distributional forms are both flexible and reasonable in describing the failure behavior during software development process. As pointed out in the above, (7) implies ordering of θ_i's for certain values of ρ and this provides flexibility in capturing various patterns for θ_i's and makes the model attractive for monitoring software reliability (see, e.g., Chen and Singpurwalla [2]). The gamma distribution assumption (3.3) is also sufficiently flexible for eliciting prior beliefs for initial failure rate θ_0.

Given (7) and (9), to develop a Bayesian updating mechanism, following Miller and Smith [12], suppose as an hypothesis

$$(\theta_{i-1} \mid X^{(i-1)}) \sim \mathrm{Gamma}(\alpha_{i-1}, \beta_{i-1}).$$

We can show by induction that the above hypothesis is true. Using (7), via distribution theory, it can be shown that the conditional distribution of $(\theta_i \mid \theta_{i-1}, X^{(i-1)})$ is a beta density with parameters $\gamma\alpha_{i-1}$ and $(1-\gamma)\alpha_{i-1}$ defined over $0 < \theta_i < \rho\theta_{i-1}$; that is,

$$p(\theta_i \mid \theta_{i-1}, X^{(i-1)}) \propto \theta_i^{\gamma\alpha_{i-1}-1}(\rho\theta_{i-1} - \theta_i)^{(1-\gamma)\alpha_{i-1}-1}.$$

Assuming that the inductive hypothesis is true, using law of total probability, it follows from the above that (see Miller and Smith [12])

$$(\theta_i \mid X^{(i-1)}) \sim \text{Gamma}\left(\gamma\alpha_{i-1}, \frac{\beta_{i-1}}{\rho}\right). \tag{10}$$

Since X_i given θ_i is exponential, using (1) and (10), the posterior distribution of θ_i given $X^{(i)}$ is obtained, via the Bayes' Theorem, as

$$(\theta_i \mid X^{(i)}) \sim \text{Gamma}(\alpha_i, \beta_i), \tag{11}$$

where $\alpha_i = \gamma\alpha_{i-1} + 1$ and $\beta_i = \frac{\beta_{i-1}}{\rho} + \text{X}_\text{i}$. Thus, the inductive hypothesis is verified.

Given the above development, using the law of total probability, the predictive distribution of X_i given $X^{(i-1)}$ is a Pareto of the form

$$p(x_i \mid X^{(i-1)}) = \frac{\gamma\alpha_{i-1}\left(\frac{\beta_{i-1}}{\rho}\right)^{\gamma\alpha_{i-1}}}{\left(\frac{\beta_{i-1}}{\rho} + \text{x}_\text{i}\right)^{\gamma\alpha_{i-1}+1}}. \tag{12}$$

A version of the above model has been used by Chen and Singpurwalla [2] for tracking reliability growth of software. As pointed out by Chen and Singpurwalla [2], the parameter ρ provides information about the growth or decay of the reliability of software. For example, the values of ρ less than or equal to 1 imply that the failure rates are decreasing from one stage to another whereas $\rho > 1$ implies that failure rates may be increasing in i. We note that all the above results are conditional on the growth parameter ρ. As noted by the above authors, it is more reasonable to treat ρ as an unknown quantity representing the behavior of failure rates over stages of testing. When ρ is unknown, as the prior distribution we define a k-point discrete distribution. We define the discrete distribution by using a discretization of the beta density on (ρ_L, ρ_U) since this allows for great flexibility in representing prior uncertainty. The beta density is given by

$$g(\rho \mid X^{(0)}) = \frac{\Gamma(c+d)}{\Gamma(c)\Gamma(d)} \frac{(\rho - \rho_L)^{c-1}(\rho_U - \rho)^{d-1}}{(\rho_U - \rho_L)^{c+d-1}} \qquad \text{for } 0 \leq \rho_L \leq \rho \leq \rho_U, \tag{13}$$

where $\rho_L, \rho_U, c, d > 0$ are specified parameters. We define our distribution for ρ as

$$p(\rho_l \mid X^{(0)}) = \Pr\{\rho = \rho_\ell \mid X^{(0)}\} = \int_{\rho_\ell - \delta/2}^{\rho_\ell + \delta/2} g(\rho)\, d\rho, \tag{14}$$

where $\rho_\ell = \rho_L + \frac{2\ell - 1}{2}\delta$ and $\delta = \frac{\rho_U - \rho_L}{k}$ for $\ell = 1, \ldots, k$. Prior to any testing, we assume that, θ_0 and ρ are independent. After i stages of testing, given $X_i = x_i$ is observed, the posterior distribution of ρ is obtained via the standard Bayesian machinery as

$$p(\rho_l \mid X^{(i)}) \propto p(\rho_l \mid X^{(i-1)})p(x_i \mid \rho_l, X^{(i-1)}), \qquad l = 1, 2, \ldots, k, \tag{15}$$

where the likelihood term $p(x_i \mid \rho_l, X^{(i-1)})$ is the predictive density given by (12). Once the posterior distribution (15) is available, the unconditional posterior distribution of θ_i can be obtained by averaging out (11) with respect to this posterior distribution.

Alternatively, we can specify a continuous distribution for ρ and use Markov chain Monte Carlo methods as in Kuo and Yang [8, 9] for computations.

One of the attractive features of the presented model is the existence of all the predictive moments of X_{i+m} and θ_{i+m} given $X^{(i)}$ for $m > 0$ analytically. As shown by Miller and Smith [12], assuming $s < \gamma\alpha_{n-1}$, for $n \geq i + 1$,

$$\mathrm{E}(\theta_{i+m}^{-s} \mid X^{(i)}) = \frac{(\beta_i)^s\Gamma(\alpha_i - s)}{\Gamma(\alpha_i)} \prod_{n=i+1}^{i+m} (\rho)^{-s} \frac{\Gamma(\alpha_{n-1})\Gamma(\gamma\alpha_{n-1} - s)}{\Gamma(\alpha_{n-1} - s)\Gamma(\gamma\alpha_{n-1})}, \tag{16}$$

and similarly

$$\mathrm{E}(X_{i+m}^{s} \mid X^{(i)}) = \Gamma(s + 1)\, \mathrm{E}(\theta_{i+m}^{-s} \mid X^{(i)}). \tag{17}$$

For example, from the above, the $m -$ step ahead predictive means are obtained as

$$\mathrm{E}(\theta_{i+m} \mid X^{(i)}) = \frac{\alpha_i}{\beta_i} (\gamma\rho)^m,$$

$$\mathrm{E}(X_{i+m} \mid X^{(i)}) = \frac{\beta_i}{(\alpha_i - 1)\rho^m} \prod_{n=i+1}^{i+m} \frac{(\alpha_{n-1} - 1)}{(\gamma\alpha_{n-1} - 1)}. \tag{18}$$

The availability of these moments enable us to obtain optimal stopping rules in Section 4. We note that the results in (18) are conditional on the parameter ρ. The unconditional m-step ahead predictive means can be obtained by averaging out (18) with respect to this posterior distribution of ρ given by (15), that is,

$$\mathrm{E}(\theta_{i+m} \mid X^{(i)}) = \mathrm{E}_\rho(\mathrm{E}(\theta_{i+m} \mid X^{(i)}, \rho)),$$

$$\mathrm{E}(X_{i+m} \mid X^{(i)}) = \mathrm{E}_\rho(\mathrm{E}(X_{i+m} \mid X^{(i)}, \rho)). \tag{19}$$

4. OPTIMALITY OF ONE-STAGE LOOK AHEAD RULE

It is reasonable to assume that $\mathscr{L}_\mathrm{T}$ is an increasing function of X_j in (5). A simple but a reasonable candidate for the loss function is

$$\mathscr{L}_\mathrm{T}(X_j) = k_\mathrm{T}X_j, \tag{20}$$

where $k_\mathrm{T} > 0$ can be interpreted as the testing cost per unit execution time. In specifying $\mathscr{L}_\mathrm{S}$, we note that an unreliable piece of software will yield a higher loss, and thus $\mathscr{L}_\mathrm{S}$ should be an increasing function of the software failure rate at the time of release. In view of this, we assume that

$$\mathscr{L}_S(\theta_{i+1}) = k_S\theta_{i+1}, \tag{21}$$

where $k_S > 0$.

For the Theorem 1 of Section 2 to be applicable, we need to show that $E(X_j \mid X^{(i)})$ is increasing in $j = i + 1, \ldots$, and $E(\theta_j \mid X^{(i)})$ is discrete convex in $j = i + 1, \ldots$. Using the model presented in Section 3, with (20) and (21) we can identify the conditions under which both of these requirements are satisfied. These are given in the following theorem.

THEOREM 2: If $\gamma\rho < 1$ and $\gamma\alpha_{n-1} > 1$ for $n \geq i + 1$, then the one-stage look ahead rule (6) of Theorem 1 is optimal.

PROOF: First we need to show that $E(X_j \mid X^{(i)})$ is increasing in j for $j \geq i + 1$. Using (18), we can write for $j \geq i + 1$

$$E(X_j \mid X^{(i)}) = \frac{\beta_i}{(\alpha_i - 1)} \prod_{n=i+1}^{j} \frac{1}{\rho}\frac{(\alpha_{n-1} - 1)}{(\gamma\alpha_{n-1} - 1)}. \tag{22}$$

We note that if the term

$$\frac{1}{\rho}\frac{(\alpha_{n-1} - 1)}{(\gamma\alpha_{n-1} - 1)} = \frac{(\alpha_{n-1} - 1)}{(\rho\gamma\alpha_{n-1} - \rho)} > 1$$

in the product then $E(X_j \mid X^{(i)})$ is increasing in j for $j \geq i + 1$, with the restriction that $\gamma\alpha_{n-1} > 1$ for $n \geq i + 1$. We note that when $\gamma\rho < 1$, this ratio is always greater than 1 guaranteeing that $E(X_j \mid X^{(i)})$ is increasing in j for $j \geq i + 1$.

To show that $E(\theta_j \mid X^{(i)})$ is discrete convex in j for $j \geq i + 1$, we can write from (16) that, for $j \geq i + 1$, the first difference is given by

$$[E(\theta_j \mid X^{(i)}) - E(\theta_{j+1} \mid X^{(i)})] = \frac{\alpha_i}{\beta_i}(\rho\gamma)^{j-i}(1 - \rho\gamma). \tag{23}$$

We note that the term on the right-hand side of (23) will be decreasing in j when $\gamma\rho < 1$.

Thus, if $\gamma\rho < 1$, then Theorem 1 is applicable and the one-stage look ahead rule is optimal. □

COROLLARY 1: When $\rho = 1$ and $\gamma\alpha_{n-1} > 1$ for $n \geq i + 1$, then the one-stage look ahead rule (6) of Theorem 1 is optimal.

PROOF: Note that $0 < \gamma < 1$ and therefore the condition $\gamma\rho < 1$ is satisfied when $\rho = 1$. □

The discrete convexity of $E(\theta_j \mid X^{(i)})$ in $j \geq i + 1$ in Theorem 2 implies that software reliability improvement diminishes during the testing/modification process. This is expected in software testing, due to the fact that the more major software bugs will be discovered and fixed during the earlier stages of the process.

If $\rho = 1$, then (7) implies that $\{\theta_i\}$ is a decreasing sequence in i since ξ_i takes values between 0 and 1. In the case $\rho > 1$, from (7) we note that $E\left(\frac{\theta_i}{\theta_{i-1}} \mid \theta_{i-1}\right) = \gamma\rho$. The condition $\gamma\rho < 1$ implies that we expect the failure rate to be decreasing from one stage to another. Thus, the one-stage look ahead rule is optimal if the failure rates are stochastically decreasing.

It is important to note that for the predictive means in (18) and (22) to exist we need to have $\gamma\alpha_{n-1} > 1$ for $n \geq i + 1$. As n gets large it can be shown that α_n goes to the *steady state value* $\alpha = \frac{1}{1-\gamma}$, and this implies that for the above condition to be satisfied we need $\frac{\gamma}{1-\gamma} > 1$ or equivalently $\gamma > 0.5$.

As we have previously noted, it is reasonable to treat ρ as unknown and describe uncertainty about ρ probabilistically as in (14). By making inference on ρ at any stage of testing, we can also assess the optimality of the one-stage look ahead rule by using probability statements. If the optimality of the rule is assessed with high probability, then we only need to compare $L_i^{(0)}$ with $L_i^{(1)}$ to determine whether to stop testing. In other words, in view of the Theorem 2, the posterior distribution of ρ is an indicator of whether the one-stage look ahead rule is optimal.

Using the results of Section 3 and the specific form of loss functions given by (20) and (21), conditional on ρ, we can obtain the additional expected loss associated with testing for δ more stages after the ith modification to the software as

$$L_i^{(\delta)} = \left\{\sum_{j=1}^{\delta} k_T \frac{\beta_i}{(\alpha_i - 1)\rho^j} \prod_{n=i+1}^{i+j} \frac{(\alpha_{n-1} - 1)}{(\gamma\alpha_{n-1} - 1)}\right\} + \left\{k_S \frac{\alpha_i}{\beta_i} (\gamma\rho)^{\delta+1}\right\}. \tag{24}$$

It follows from (24) that

$$L_i^{(0)} = \left\{k_S \frac{\alpha_i}{\beta_i} (\gamma\rho)\right\},$$

$$L_i^{(1)} = \left\{k_T \frac{\beta_i}{(\gamma\alpha_i - 1)\rho} + k_S \frac{\alpha_i}{\beta_i} (\gamma\rho)^2\right\}. \tag{25}$$

By using the posterior distribution of ρ given by (15), we can average out (24) and obtain $L_i^{(\delta)}$ unconditional on ρ.

4.1. Elicitation of Loss Function and Prior Distribution Parameters

The Bayesian decision theoretic formulation of the optimal stopping problem requires specification of loss function parameters as well as the parameters of the prior distributions.

In our setup the loss function parameters are directly associated with costs of testing and the costs of releasing an unreliable piece of software. In (20) k_T can be interpreted as testing cost per unit time and it can be estimated as a financial cost. Relative to k_S, k_T is expected to be very low in many applications. In (21) it is expected that an unreliable software will yield a high loss if it is released and thus we expect k_S to be high. Typically k_S will reflect costs associated with discovering and fixing bugs during usage phase as well as any replacement type costs. In many applications it may be possible to estimate k_S as a multiple of k_T (see McDaid and Wilson [11]).

In the model presented in Section 3, we need to specify prior parameters α_0 and β_0 for the distribution of θ_0, parameters ρ_L, ρ_U, c, and d for the distribution of ρ and the value of parameter γ.

We note that θ_0 represents the failure rate of the software prior to any testing and it is proportional to the initial number of bugs present in the software. Higher values of θ_0 imply longer testing periods for the software. In specifying the prior distribution of θ_0 given by (9), we can elicit a *best guess* value for θ_0 and an uncertainty measure about this best guess from the software developer (or the software engineer). This best guess value can be considered as the mean estimate for the gamma distribution; that is, it can be equated to α_0/β_0 [alternatively it can be treated as the mode estimate and can be equated to $(\alpha_0 - 1)/\beta_0$]. The elicited uncertainty measure can be considered as the spread or the standard deviation and it can be equated to $\sqrt{\alpha_0}/\beta_0$. Given the mean and standard deviation estimates from the developer, the values of α_0 and β_0 can be solved from the mean and standard deviation identities.

As previously discussed, the distribution of ρ plays an important rule in optimality of the one-stage look ahead rule. Also, together with γ, ρ represents our perceptions of the performance of debugging process. We note that if the prior for ρ is degenerate at 1, that is, $\rho = 1$, then this implies that the software always improves with testing. Thus, the value 1 is a reasonable lower bound for ρ, and we can choose $\rho_L = 1$. On the other hand, if $\Pr(\gamma\rho > 1 \mid X^{(0)}) = 1$, then there will be no improvement in software's failure rate as a result of additional testing and this would imply that given such a prior information the software will not be tested at all. This can be seen from Eq. (23) by letting $j = 1$ and $i = 0$. Note that when $\gamma\rho > 1$ with probability 1 as a result of additional testing, we expect that the failure rate will increase. Thus, very large values are not reasonable for ρ_U. Once the value of γ is fixed, an upper bound and a lower bound (typically 1) can be elicited from the software developer to specify ρ_L and ρ_U. Once these are given, a best guess value can be elicited for ρ. This can be treated as the mean estimate and is equated to $\rho_L + (\rho_U - \rho_L)c/(c + d)$, the mean of the beta density. Similarly, a spread estimate around this best guess can be elicited and equated to the standard deviation of the beta density. These expressions can then be solved to obtain c and d. Other elicitation procedures have been proposed for the beta distribution by Chaloner and Duncan [1].

In eliciting the parameters of the prior distribution of ρ and the value of γ, forms of expressing the performance of debugging process may facilitate expert input. For example, we note that $100 \times [1 - \gamma \mathrm{E}(\rho \mid X^{(0)})]$ represents the percentile expected improvement (that is, decrease) in failure rate as a result of one additional period of testing. If such information can be elicited, then it can be used in specifying the value of γ as well as the prior parameters of the distribution of ρ.

It is important to note that the decision of when to stop testing may be sensitive to the choice of above parameters. Thus, it is always desirable to investigate the sensitivity of the results to the choice of prior parameters and loss function components. As will be seen in Section 5.2, such sensitivity analysis can be easily performed.

5. ILLUSTRATION OF STOPPING RULES

In this section we will illustrate the use of stopping rules using two examples one based on hypothetical and the other based on real data.

5.1. Hypothetical Data Example

Assume that a particular piece software will go through at most 10 stages of testing/debugging, and after each stage, following the modifications made to the software, a decision is made whether to terminate the testing process.

Table 1. Expected additional loss estimates after each stage.

Stage j	0	1	2	3	4	5	6	7	8	9	10
$L_0^{(j)}$	80,000	64,300	51,200	40,900	32,800	26,200	21,000	16,800	13,500	10,800	8700
$L_1^{(j-1)}$	—	41,600	33,300	26,600	21,300	17,100	13,700	10,900	8,800	7100	5700
$L_2^{(j-2)}$	—	—	7100	5700	4600	3700	3000	2500	2100	1800	1700
$L_3^{(j-3)}$	—	—	—	1800	1600	1400	1300	1200	1300	1400	1700
$L_4^{(j-4)}$	—	—	—	—	1300	1200	1100	1000	1100	1300	1500
$L_5^{(j-5)}$	—	—	—	—	—	930	900	910	990	1200	1500
$L_6^{(j-6)}$	—	—	—	—	—	—	950	910	900	990	1200
$L_7^{(j-7)}$	—	—	—	—	—	—	—	980	930	910	980
$L_8^{(j-8)}$	—	—	—	—	—	—	—	—	810*	820	860

Case 1, ρ is known: Assume that, prior to any testing, uncertainty about software's performance is expressed by the model given in (7) with $\rho = 1$, $\gamma = 0.8$, and the prior parameters $\alpha_0 = \beta_0 = 2$. Thus, our model implies that the failure rates will be decreasing with each stage of testing. This assumption will be dropped in the second part of our analysis. We also assume that $k_T = 1$ and $k_S = 100{,}000$ in (20) and (21) implying a relatively high loss of releasing an unreliable piece of software. We note that, as shown in Section 4, when $\rho = 1$ the one-stage look ahead rule is optimal.

Based on the prior information alone the $L_0^{(\delta)}$ can be obtained using (24). These values are presented in the first row of Table 1 for $\delta = 0, 1, \ldots, 10$. Using the stopping rule (6), $L_0^{(1)} - L_0^{(0)} < 0$ and thus the optimal decision is to initiate testing. Furthermore, using (4), $L_0^* = L_0^{(10)}$, which implies that testing is expected to terminate after 10 stages.

Suppose that software has failed after 3 units of time during the first testing stage. After the completion of the first stage, uncertainty about the software failure rate is revised using the results of Section 3, and the expected additional loss is calculated using (24). This is given in the second row of Table 1. Again as $L_1^{(1)} - L_1^{(0)} < 0$, the optimal decision is to continue testing. Since $L_1^* = L_1^{(9)}$, the testing is still expected to be completed after the 10th stage.

The observed life-lengths of the software at the subsequent testing stages are given in Table 2, and the corresponding expected additional losses, $L_i^{(j-i)}$, are given in Table 1. Note that the optimal decision rule is to stop testing the first time $L_i^{(1)} > L_i^{(0)}$. We see from Table 1 that the optimal decision after testing stages 2–7 is to continue testing. It is not until the 8th stage that $L_8^{(1)} > L_8^{(0)}$ implying that the optimal decision is to stop testing.

In Figure 2, we illustrate the posterior means of the failure rates, θ_i's after each stage of testing. The downward trend in θ_i's reflects the improvement in the performance of the software as implied by the model. The sequential nature of the inference and decision making during the testing/debugging process is shown in Figure 3, where we present the plots of expected additional loss after 0, 3, 6, and 8 stages of testing. These curves are obtained from the rows 1, 4, 7, and 9 of Table 1. The plot shows how the expected duration of the testing changes with additional observations. For example, apriori testing is expected to terminate after the 10th stage. After 3 stages of testing, the expected termination is revised to the 7th stage, and after 6 stages of testing it is revised to the 8th stage.

Table 2. Actual life-length of software tested in stages $j = 1, \ldots, 8$.

Testing Stage j	1	2	3	4	5	6	7	8
X_j	3	30	113	81	9	2	91	112

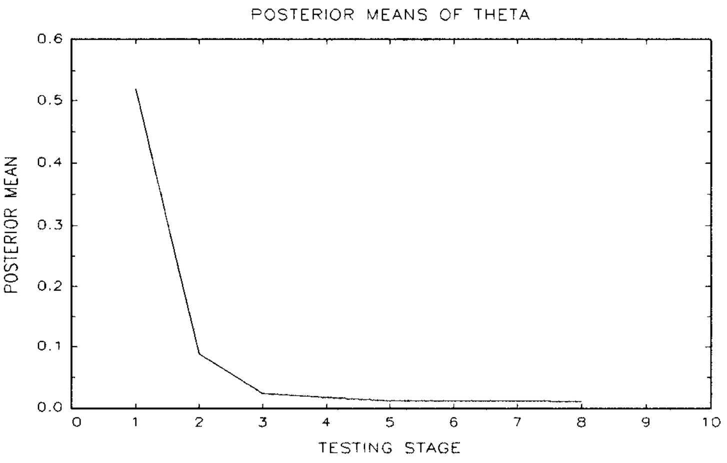


Figure 2. Posterior means of θ_i's.

Case 2, unknown ρ: We now assume that ρ is unknown in (7), and we describe our uncertainty about ρ *a priori* via the discrete beta density given by (13). We specify $\rho_L = 1$, $\rho_U = 2$, $c = 1.25$, and $d = 5$ in (13). Such a choice reflects the prior belief of expected improvement in software's performance. As in part 1 of our analysis, we choose $\gamma = 0.8$, $\alpha_0 = \beta_0 = 2$ and assume that $k_T = 1$ and $k_S = 100{,}000$. We note that such a choice of prior parameters does not guarantee that the one-stage look ahead rule is optimal since $\gamma\rho > 1$ for $\rho > 1.25$.

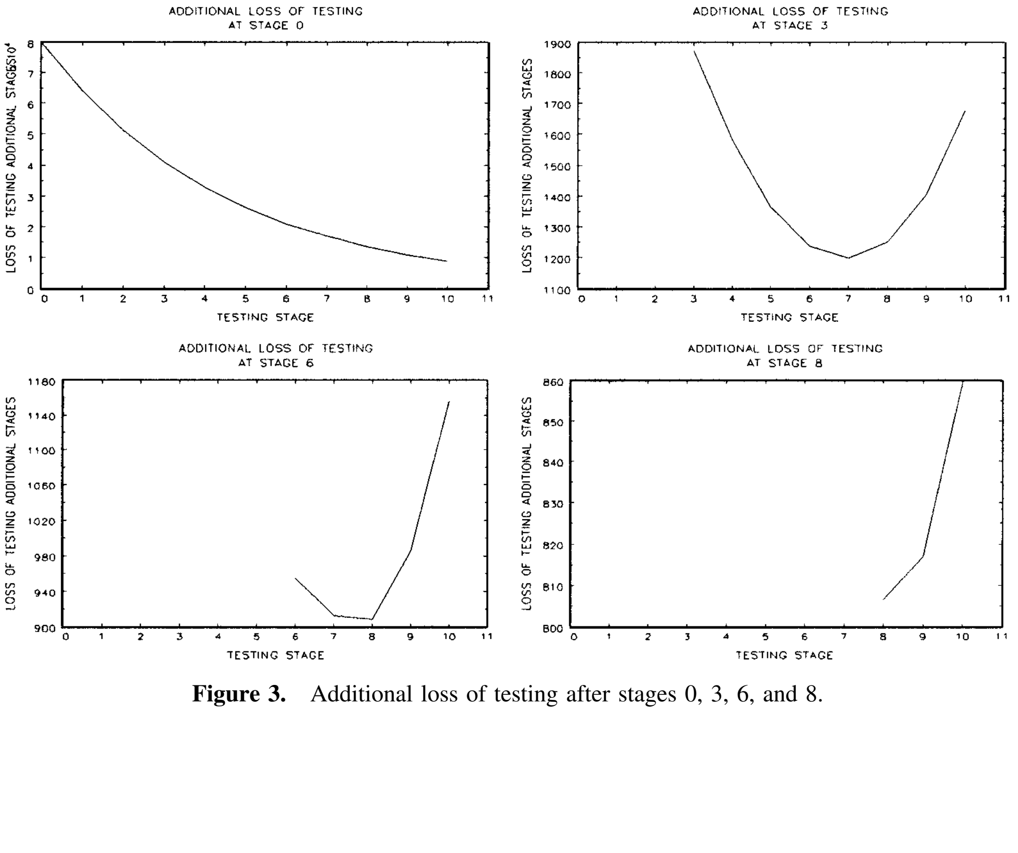


Figure 3. Additional loss of testing after stages 0, 3, 6, and 8.

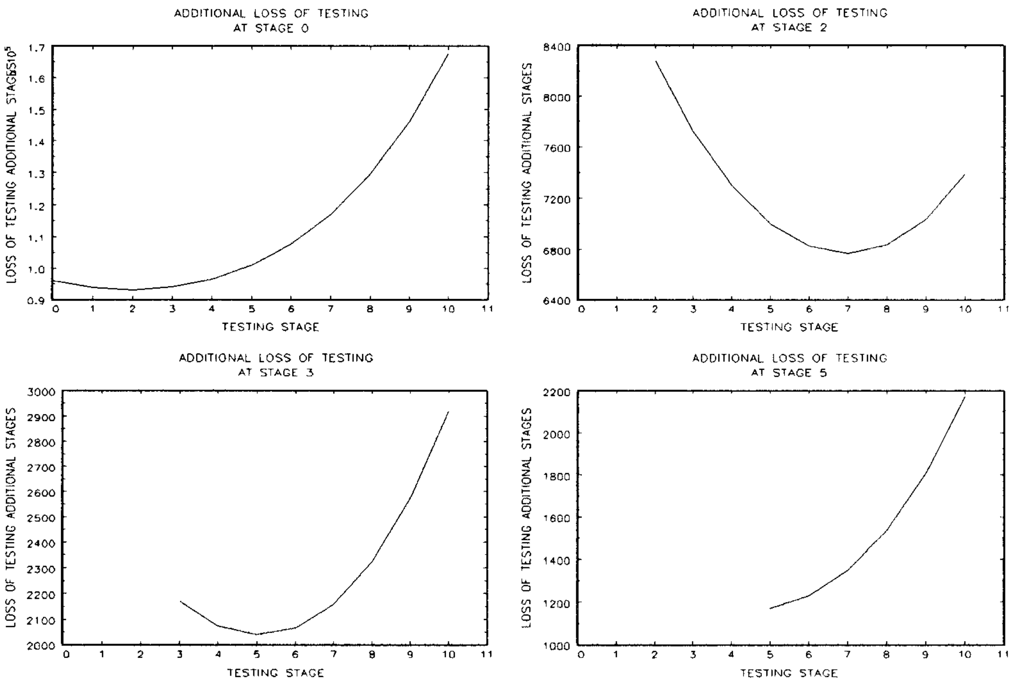


Figure 4. Additional loss of testing after stages 0, 2, 3, and 5.

In Figure 4 we present the plots of expected additional loss after 0, 2, 3, and 5 stages of testing. In this particular case it is optimal to stop after 5 stages of testing. We can see from the plot that the one-stage look ahead rule is optimal in all the four cases. As in part 1 of the analysis this plot shows how the expected duration of the testing changes with additional observations. For example, *a priori* we expect to terminate the testing after two stages. After the 2nd stage, we revise the expected termination to stage 7 and after the 3rd stage we revise it to stage 5, which was the optimal stage to terminate testing.

Figure 5 shows the prior and the posterior distributions of ρ after 2, 3, and 5 stages of testing. We can see from the figure that the posterior mass shifts more to the left (closer to 1) with additional observations implying an expected improvement in software. We can compute $\Pr\{\rho < 1.25 \mid X^{(i)}\}$, the probability that $\rho < 1.25$ after i stages of testing, for any stage i. For example, with the particular values of prior parameters, we obtain $\Pr\{\rho < 1.25 \mid X^{(5)}\} = 0.915$, reinforcing the optimality of the one-stage look-ahead rule.

5.2. Real Data Example

In this section we consider the naval tactical data system (NTDS) data published in Jelinski and Moranda [7]. The data is given in Table 3, where X_j represents the actual life-length of software at testing stage j. This data was also used by McDaid and Wilson [11] to illustrate testing strategies using nonhomogeneous Poisson process type models. In their illustration, using the bug-counting models, the authors found out that it was optimal to stop after 23 stages of testing.

In our analysis we describe our prior uncertainty about ρ using the discrete beta distribution of (13) with $\rho_L = 1$, $\rho_U = 2$, $c = 1$, and $d = 7$. This choice prior parameters implies that

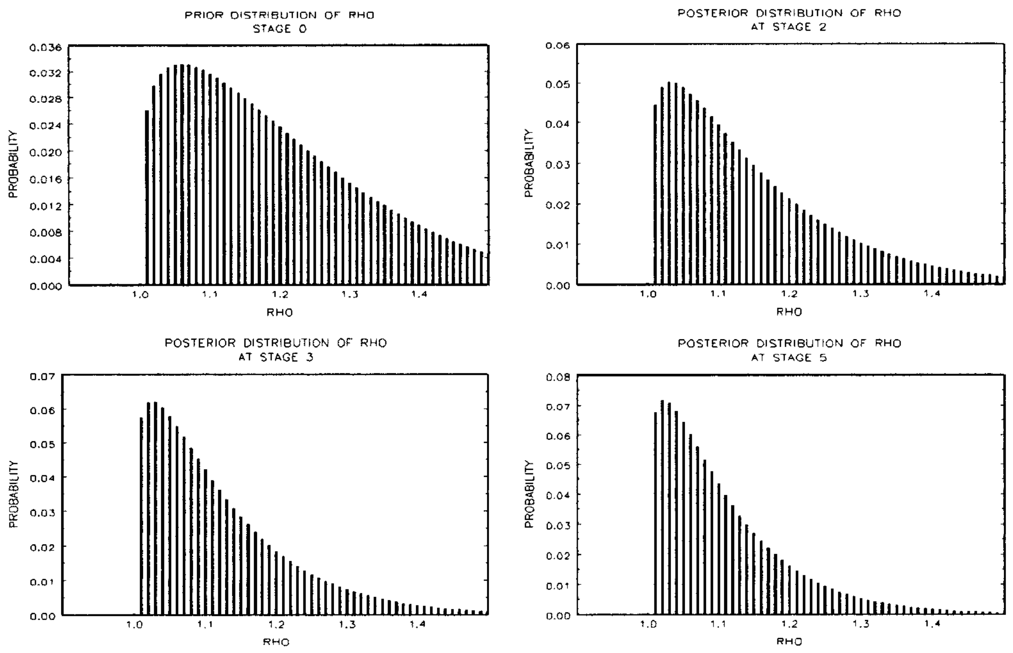


Figure 5. Prior and posterior distributions of ρ after testing stages 0, 2, 3, and 5.

$E(\rho \mid X^{(0)}) = 1.125$. Furthermore, we specify $\gamma = 0.8$ to reflect a prior belief of expected improvement in software's performance with $\Pr(\gamma\rho < 1 \mid X^{(0)}) = 0.865$. As in Section 5.1, we assume that $k_T = 1$ and $k_S = 100{,}000$. The prior parameters of θ_0 are chosen as $\alpha_0 = 5$ and $\beta_0 = 2$. We note that $\alpha_0 = 5$ is the steady-state value for α_i with $\gamma = 0.8$.

The above selection of parameters imply that optimal stopping time is after 20 stages of testing. We present in Figure 6 the plots of expected additional loss after 0, 7, 14, and 20 stages

Table 3. NTDS data of Jelinski and Moranda [7].

Testing Stage j	X_j	Testing Stage j	X_j
1	9	18	3
2	12	19	6
3	11	20	1
4	4	21	11
5	7	22	33
6	2	23	7
7	5	24	91
8	8	25	2
9	5	26	1
10	7	27	87
11	1	28	47
12	6	29	12
13	1	30	9
14	9	31	135
15	4	32	258
16	1	33	16
17	3	34	35

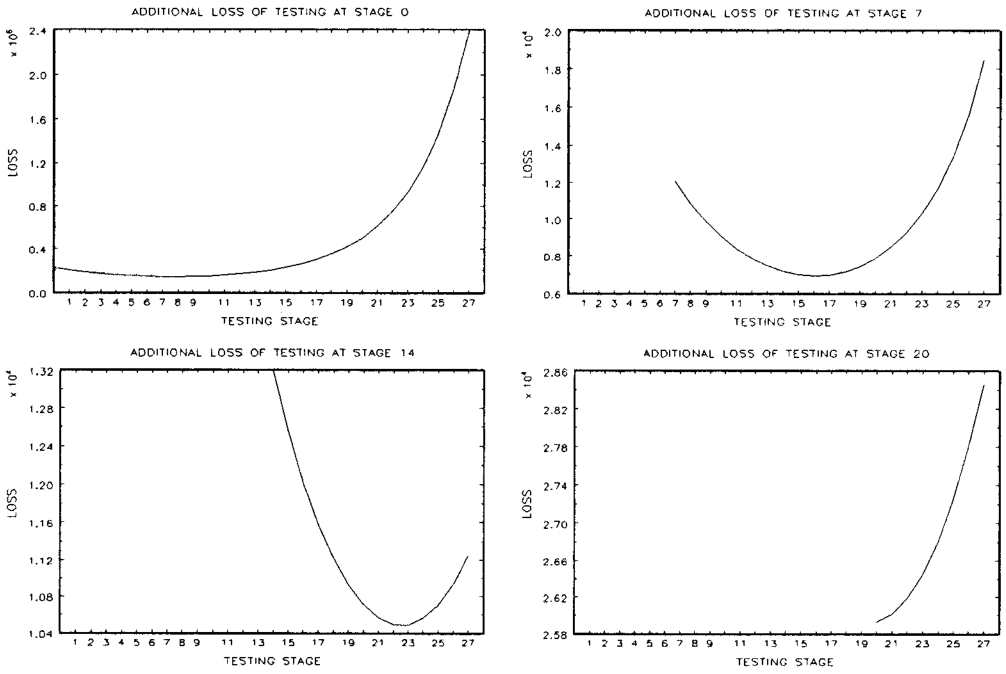


Figure 6. Additional loss of testing after stages 0, 7, 14, and 20.

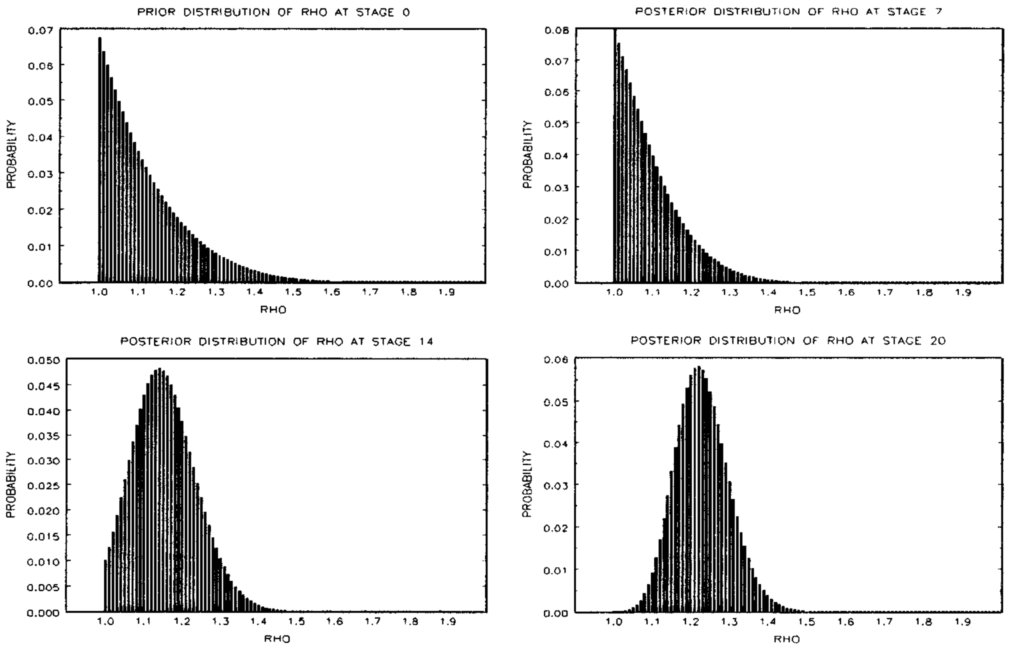


Figure 7. Prior and posterior distributions of ρ after testing stages 0, 7, 14, and 20.

Table 4. Sensitivity to loss parameter k_S.

k_S	100,000	25,000	10,000	2500	1000
Optimal stopping stage	20	20	18	13	5

of testing. We can see from the plots that the one-stage look-ahead rule is optimal in all the four cases. As before, we can see from the plot how the expected duration of testing changes with new data. Prior to any testing it is expected that testing will be terminated after 8 stages. After 7 stages of testing, the expected termination is revised to 16 stages, and finally it is decided that testing is stopped after 20 stages.

In Figure 7 we present the prior and posterior distributions of ρ after 7, 14, and 20 stages. We note that the posterior mass shifts away from the value 1 with additional testing. The posterior probability $\Pr(\gamma\rho < 1 \mid X^{(i)})$ reaches 0.938 after 7 stages of testing, but it reduces to 0.609 after 20 periods implying that expected improvement from additional testing is less likely.

As previously mentioned, we can investigate the sensitivity of our results to the choice parameters. In so doing, we will be interested in how the optimal stopping time changes with different parameter specifications.

We expect that decreases in k_S will yield an earlier optimal stopping time. By keeping the same specifications for all other parameters, we have repeated our analysis for values of k_S = 25,000, 10,000, 2500, 1000. The corresponding optimal stopping times are presented in Table 4. We note that for a considerable decrease in optimal stopping time, a significant drop in k_S is necessary in this case.

We have also investigated how the optimal stopping time changes with the prior parameters of the distribution of ρ. In so doing, we have kept the same values for prior parameters ρ_L, ρ_U, and $c = 1$ and repeated the analysis for $d = 4$, 5, and 9. We note that decreasing the value of d implies a higher value for the prior mean and the variance of ρ. In this case the choice of $d = 5$ decreased the optimal stopping period to 16 and the choice of $d = 4$ implied optimal stopping after 13 stages. Increasing the value of d to 9 resulted in an optimal stopping stage of 32.

Our analyses using different values of α_0 and β_0 suggest that optimal stopping time is more sensitive to the choice of α_0 then to β_0. As expected, increases in the value of α_0 imply an increase optimal stopping time whereas increase in β_0 yields earlier stopping.

We note that another important parameter in the analysis is γ. As previously pointed out γ is expected to take values between 0.5 and 1. We have repeated our analysis for different values of γ by keeping everything else constant. Our findings that are presented in Table 5 suggest that optimal stopping time is sensitive to changes in γ. Thus, it is important to elicit this parameter accurately.

Alternatively a prior distribution can be specified for γ and uncertainty about γ can be revised via Bayesian machinery as test information becomes available. To investigate what values of γ are appropriate for NTDS data, we have assumed a discrete uniform prior for γ over $0.5 < \gamma < 1$ independent of all other parameters and obtained its posterior distribution. In Table 6 we present the posterior means of γ and ρ after various stages of testing. Based on the table, it looks

Table 5. Sensitivity to γ.

γ	0.75	0.80	0.85	0.90
Optimal stopping stage	31	20	18	0

Table 6. Posterior means of γ and ρ.

j	5	10	15	20	25	30	34
$E(\gamma \mid X^{(j)})$	0.63	0.75	0.82	0.85	0.78	0.80	0.79
$E(\rho \mid X^{(j)})$	1.10	1.14	1.15	1.15	1.18	1.17	1.17

like the posterior mean of γ stabilizes around 0.80–0.85 after 15 stages of testing. Thus, our choice of $\gamma = 0.80$ seems quite appropriate for this data.

ACKNOWLEDGMENTS

The authors are grateful to the associate editor and the two referees for their suggestions which led to an improved exposition.

REFERENCES

[1] K.M. Chaloner and G.T. Duncan, Assessment of a beta prior distribution: PM elicitation, Statistician 32 (1983), 174–180.
[2] Y. Chen and N.D. Singpurwalla, A non-Gaussian Kalman filter model for tracking software reliability, Stat Sin 4 (1994), 535–548.
[3] S.R. Dalal and C.L. Mallows, When to stop testing software? J Am Stat Assoc 83 (1986), 872–879.
[4] A. Erkanli, T.A. Mazzuchi, and R. Soyer, Bayesian computations for a class of reliability growth models, Technometrics 40 (1998), 14–23.
[5] E.H. Forman and N.D. Singpurwalla, An empirical stopping rule for debugging and testing computer software, J Am Stat Assoc 72 (1977), 750–757.
[6] E.H. Forman and N.D. Singpurwalla, Optimal time intervals for testing hypotheses on computer software errors, IEEE Trans Reliab 28 (1979), 250–253.
[7] Z. Jelinski and P. Moranda, "Software reliability research," Statistical computer performance evaluation, W. Freiberger (Editor), Academic, New York, pp. 465–484.
[8] L. Kuo and T.Y. Yang, Bayesian computation of software reliability, J Comput Graph Stat 4 (1995), 65–82.
[9] L. Kuo and T.Y. Yang, Bayesian computation for nonhomogeneous Poisson processes in software reliability, J Am Stat Assoc 91 (1996), 763–773.
[10] D.V. Lindley, The present position in Bayesian statistics, Stat Sci 5 (1990), 44–89.
[11] K. McDaid and S.P. Wilson, Deciding how long to test software, Statistician 50 (2001), 117–134.
[12] J.E. Miller and R.L. Smith, A non-Gaussian state space model and application to prediction of records, J Roy Stat Soc Ser B 48 (1986), 79–88.
[13] K. Okumoto and A.L. Goel, Optimum release time for software systems, based on reliability and cost criteria, J Syst Software 1 (1980), 315–318.
[14] S. Ozekici and N.A. Catkan, A dynamic software release model, Comput Econ 6 (1993), 77–94.
[15] S.M. Ross, Software reliability: The stopping rule problem, IEEE Trans Software Eng 11 (1985), 1472–1476.
[16] N.D. Singpurwalla, "Preposterior analysis in software testing," Statistical data analysis and inference, Y. Dodge (Editor), Elsevier, Amsterdam, 1989, pp. 581–595.
[17] N.D. Singpurwalla, Determining an optimal time interval for testing and debugging software, IEEE Trans Software Eng 17 (1991), 313–319.
[18] J.R. van Dorp, T.A. Mazzuchi, and R. Soyer, Sequential inference and decision making during product development, J Stat Plan Inference 62 (1997), 207–218.
[19] S. Yamada, H. Narihisa, and S. Osaki, Optimum release policies for a software system with a scheduled software delivery time, Int J Systems Science 15 (1984), 905–914.