Bayesian Testing Strategies for Software with an Operational Profile

S. Özekici,[1] R. Soyer[2]

[1]*Department of Industrial Engineering, Boğaziçi University, 80815 Bebek-Istanbul, Turkey*

[2]*Department of Management Science, The George Washington University, Washington, DC 20052*

Received July 2000; revised March 2001; accepted 28 May 2001

Abstract: We consider a software reliability model where the failure rate of each fault depends on the specific operation performed. The software is tested in a given sequence of test cases for fixed durations of time to collect data on failure times. We present a Bayesian analysis of software failure data by treating the initial number of faults as a random variable. Our analysis relies on the Markov Chain Monte Carlo methods and is used for developing optimal testing strategies in an adaptive manner. Two different models involving individual and common faults are analyzed. We illustrate an implementation of our approach by using some simulated failure data. © 2001 John Wiley & Sons, Inc. Naval Research Logistics 48: 747–763, 2001

Keywords: Software testing; operational profile; statistical inference; Bayesian analysis

1. INTRODUCTION

In this article, we discuss some interesting statistical issues that arise in the testing of software with an operational profile. An operational profile, as described by Musa [9, 10], consists of the set of all possible operations that a system is designed to perform and their occurrence probabilities. It quantifies, in a probabilistic sense, how a software will be used in the field. The main contribution of this profile in software reliability engineering is that it reduces system risk by making it more realistic. Moreover, it makes testing faster and more efficient. It is this aspect of the software development cycle that we will focus on in this article. To put it more precisely, we consider the testing phase where the software is tested sequentially for given durations of time in a number of test cases.

Özekici and Soyer [12] recently introduced a stochastic model of software failures when the operational process, in fact, is a Markov process. Wohlin and Runeson [19] also discuss the effect of usage modelling in software certification. A stochastic model of software usage involving Markov chains is employed in Whittaker and Poore [17] and Whittaker and Thomason [18]. In their approach, the sequence of ''inputs'' provided by the user is modeled as a Markov chain. This results in a model at the micro level involving all possible values of input variables with a huge state

Correspondence to: S. Özekici

space. An operational process, on the other hand, provides a stochastic model at a more refined macro level because an operation corresponds to a specific task which usually involves ranges of values for many input variables at the same time. The operational profile model concentrates on the user-initiated tasks performed by the system rather than the sequence of user-supplied input values. In the present setting, however, our aim is to consider several statistical issues given the data observed in a controlled testing environment. For a complete discussion on the stochastic foundations for describing and incorporating the operational profile, the reader is referred to Singpurwalla et al. [16].

We assume that there are K distinct operations performed in L test cases, and the duration of testing in each case is fixed in advance. Testing is done in a sequential manner, and, during each test case, the number of failures as well as their failure times are observed. As soon as a failure is observed, the fault that caused it is removed with certainty after debugging. In other words perfect debugging is assumed. A static decision making problem with exponential failure times and perfect debugging is discussed in Özekici, Altınel, and Özçelikyürek [13]. In a recent article, Özekici, Altınel, and Angün [14] analyze a model with arbitrary failure time distribution and imperfect debugging. These articles, as well as most others in the literature, assume that all model parameters are known in advance. Our primary objective in this paper is to extend this line of research in a Bayesian framework where the parameters are not necessarily known. The Bayesian framework provides learning about unknown parameters as testing is done and thus enables us to revise our testing strategies in a dynamic manner.

In Section 2, we will present the model for software failures recently introduced by Özekici and Soyer [12]. Then, we will discuss the Bayesian analysis of the model using Markov Chain Monte Carlo methods or, more precisely, the Gibbs sampler. Our discussion will be on the individual faults model in Section 3, and the common faults model in Section 4. A special important case of the common faults model is discussed in Section 5. The Bayesian approach will be used to develop dynamic testing strategies in Section 6, and its implementation will be demonstrated by an example in Section 7.

2. MODEL FOR SOFTWARE FAILURES

A Markov process model is considered by Özekici and Soyer [12] to describe the failure behavior of software with an operational profile. The model implies that, after release, the sequence of operations performed is a Markov chain and the amount of time spent on each operation is exponentially distributed. This model can be considered as an extension of the Jelinski and Moranda [5] model where the failure rate is a function of the operation performed by the software. However, before release, during the software development phase, testing is done in a prescribed sequential manner using L different test cases with constant duration τ_l for test case l. Testing strategies for a software with an operational profile are recently discussed by Özekici, Altınel, and Özçelikyürek [13]. The authors point out that the testing cases may require the use of all or some of the K operations in the software and consider two plausible testing scenarios. Under the first scenario, it is assumed that during test case k only a particular type of fault, say, type k, can cause the failure of the software. In other words, each fault is associated with one operation. This model is referred to as the *individual faults* model by the authors. The second scenario assumes that during test case k any type of fault can cause the failure of the software. This model is referred to as the *common faults* model.

The model that should be used depends on the design of the software. If, for example, the design involves a modular structure without interaction between the modules, then it may be more appropriate to use the *individual faults* model. This is true in particular if each operation

executes a specific module and causes failures due to possible faults within that module. The opposite is true for the *common faults* model since each operation can access any one fault in the whole software. In a more general setup, there may be mixed models where both faults and operations are coclassified such that operations can access variable subsets of the faults. Each operation will then be associated with some types of faults which do not necessarily include all types at the same time (*common faults* model) or only one (*individual faults* model). Such a model will require further classification of the faults with respect to a set of fault types and an accessibility relationship between each operation and fault type. The *individual* and *common faults* models considered in this article provide two scenarios with sufficient interest, applicability and computational tractability.

Both models assume that the failure rate of each fault is a function of both the test case used and the operation performed by the software. Thus, during test case l, the failure rate of each fault due to operation k will be denoted by λ_{lk}. Özekici, Altınel, and Özçelikyürek [13] assume that time between failures of the software is exponentially distributed and that the debugging is perfect. In other words, it is assumed that after a failure is observed the fault causing the failure can be identified and removed. This is also the case in the model proposed by Özekici and Soyer [12] for describing software failures with an operational profile.

Let $U_l(m)$ denote the time (since the start of the test case l) of the mth failure during test case l. Then, under the *individual faults* model, the conditional distribution of the time to the next failure of the software satisfies

$$P[U_l(m) - U_l(m-1) > t | N_l^k(m); k = 1, 2, \ldots, K] = \prod_{k=1}^{K} e^{-N_l^k(m)\lambda_{lk}t}, \tag{1}$$

where $N_l^k(m)$ denotes the number of type k faults remaining in the software before the mth failure during test l. Since testing is done sequentially, $N^k = N_1^k(1)$ denotes the initial number of type k faults in the software before any testing. The initial number of faults $N^k, k = 1, \ldots, K$, are assumed to be independent random quantities each with the Poisson distribution

$$P[N^k = m] = \frac{e^{-\mu_k}\mu_k^m}{m!} \tag{2}$$

for $m = 0, 1, \ldots$, where μ_k is the expected number of faults of type k prior to any testing.

Under the *common faults* model, the conditional distribution of the time to the next failure caused by any common fault during test case l now satisfies

$$P[U_l(m) - U_l(m-1) > t | N_l] = \prod_{k=1}^{K} e^{-(N_l - m + 1)\lambda_{lk}t} = e^{-(N_l - m + 1)\lambda_l t}, \tag{3}$$

where N_l denotes the number of common faults remaining in the software before test case l and $\lambda_l = \sum_{k=1}^{K} \lambda_{lk}$ is the total failure rate during test case l due to all operations. Thus, $N = N_1$ denotes the initial number of faults in the software prior to any testing. The initial number of faults N is assumed to be a Poisson random variable with mean μ, that is,

$$P[N = m] = \frac{e^{-\mu}\mu^m}{m!} \tag{4}$$

for $m = 0, 1, \ldots$.

Under both models statistical issues of interest include predicting the remaining number of faults in the software and assessment of the reliability of the software. These play a key role in determining the optimal release time of the software.

3. THE INDIVIDUAL FAULTS MODEL

Recall that this is the model where the set of all faults can be partitioned with respect to the operations that can access them. Any one fault can cause the failure of one operation only. Those faults that can be accessed by operation k will be referred to as type k faults. The total number of faults in the software is the sum over k of all type k faults. We can also write $\mu = \mu_1 + \mu_2 + \cdots + \mu_K$ to represent this relationship in expectation. During test case l, M_l failures are observed at times $U_l(1), \ldots, U_l(M_l)$ as well as the types of the faults causing them associated with the operations. Let M_l^k denote the number of failures caused by type k faults observed during test l such that $M_l = \sum_{k=1}^{K} M_l^k$ is the total number of failures observed during test case l.

Given data $\mathcal{D} = \{U_l(1), \ldots, U_l(M_l), M_l^k, k = 1, \ldots, K, l = 1, \ldots, L\}$ observed from L test cases performed in sequence, using K operations, the likelihood function of $\mathbf{N} = (N^1, \ldots, N^K)$ and $\boldsymbol{\lambda} = (\lambda_{11}, \ldots, \lambda_{1K}, \ldots, \lambda_{L1}, \ldots, \lambda_{LK})$ is given by

$$L(\mathbf{N}, \boldsymbol{\lambda}|\mathcal{D}) = \prod_{l=1}^{L} \prod_{k=1}^{K} \frac{N_l^k!}{(N_l^k - M_l^k)!} \lambda_{lk}^{M_l^k} \times e^{-\lambda_{lk}\{\sum_{m=1}^{M_l} N_l^k(m)[U_l(m) - U_l(m-1)] + (N^k - \sum_{j=1}^{l} M_j^k)[\tau_l - U_l(M_l)]\}}, \quad (5)$$

where $N_l^k = N_l^k(1)$ is the number of type k faults remaining before test l. If we let $M_l^k(m)$ denote the number of failures of type k observed before the mth failure during test l, then it follows from the notation of Section 2 that $N_l^k(m)$, the number of type k faults remaining in the software before the mth failure during test l, can be written as

$$N_l^k(m) = N^k - \sum_{j=1}^{l-1} M_j^k - M_l^k(m) \quad (6)$$

in the likelihood function $L(\mathbf{N}, \boldsymbol{\lambda}|\mathcal{D})$.

We assume independent gamma priors on each λ_{lk} with shape parameter a_{lk} and scale parameter b_{lk}, denoted as

$$\lambda_{lk} \sim \text{Gamma } (a_{lk}, b_{lk}) \quad (7)$$

for all $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, K$. As stated in Section 2, prior for $\mathbf{N} = (N^1, \ldots, N^K)$ is a product of K independent Poissons each with rate μ_k, denoted as $N^k \sim \text{Poisson } (\mu_k)$, and is independent of $\boldsymbol{\lambda}$. The joint posterior distribution of $\boldsymbol{\lambda}$ and $\mathbf{N}$ is given via

$$p(\boldsymbol{\lambda}, \mathbf{N}|\mathcal{D}) \propto L(\boldsymbol{\lambda}, \mathbf{N}|\mathcal{D}) p(\mathbf{N}) p(\boldsymbol{\lambda}), \quad (8)$$

where $p(\mathbf{N})$ and $p(\boldsymbol{\lambda})$ denote the prior distributions with $p(\mathbf{N})$ obtained as the product of K Poissons and $p(\boldsymbol{\lambda})$ obtained as the product of LK gamma densities. No analytically tractable posterior forms can be obtained for $\boldsymbol{\lambda}$ and $\mathbf{N}$ due to infinite sums involved in the terms, but a fully Bayesian analysis can be made using a Gibbs sampler for the posterior computations.

The implementation of the Gibbs sampler requires the full posterior conditionals $p(N^k|N^{(-k)}, \boldsymbol{\lambda}, \mathcal{D})$ and $p(\lambda_{lk}|\lambda^{(-lk)}, \mathbf{N}, \mathcal{D})$ for $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, K$, where $N^{(-k)} = \{N_i; i \neq k, i = 1, 2, \ldots, K\}$ and $\lambda^{(-lk)} = \{\lambda_{ij}; (i, j) \neq (l, k), i = 1, 2, \ldots, L \text{ and } j = 1, 2, \ldots, K\}$.

The full conditional of λ_{lk} can be obtained as

$$(\lambda_{lk}|\lambda^{(-lk)}, \mathbf{N}, \mathcal{D}) \sim \text{Gamma } (M_l^k + a_{lk}, \bar{b}_{lk}), \tag{9}$$

where $\bar{b}_{lk}$ and the details of the development are given in the Appendix. It is important to note that, given N, λ_{lk} is independent of $\lambda^{(-lk)}$ for $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, K$. In obtaining the full conditional distributions $p(N^k|N^{(-k)}, \boldsymbol{\lambda}, \mathcal{D})$, it can be shown that (see the Appendix for details)

$$(N^k - M^k|N^{(-k)}, \boldsymbol{\lambda}, \mathcal{D}) \sim \text{Poisson } (\mu_k e^{-\sum_{l=1}^{L} \lambda_{lk}\tau_l}), \tag{10}$$

where $M^k = \sum_{l=1}^{L} M_l^k$. We note that, given $(\lambda_{1k}, \ldots, \lambda_{Lk})$ and the data, N^k is independent of $N^{(-k)}$.

Thus, all of the posterior distributions can be evaluated by recursively simulating from the full conditionals in a straightforward manner. It is important to note that using the independent priors, given $\mathbf{N}$, *a posteriori* the λ_{lk}'s are independent and similarly given $\boldsymbol{\lambda}$, N^k is independent of $N^{(-k)}$. However, unconditionally, there is posterior dependence. The attractive feature of the Gibbs sampler is that uncertainty statements can be made about any λ_{lk} and N^k either sequentially or after observing data from all of the operations.

4. THE COMMON FAULTS MODEL

This is the model when all operations can access all of the faults. This relationship can be expressed as $\mu = \mu_1 = \mu_2 = \cdots = \mu_K$. During test case l, any fault can cause the failure of operation k with rate λ_{lk} and the total failure rate is $\lambda_l = \sum_{k=1}^{K} \lambda_{lk}$. Once again, M_l failures are observed at times $U_l(1), \ldots, U_l(M_l)$ as well as the operations performed at the time of the failures. In this case, let M_l^k denote the number of failures observed when operation k is performed during test l such that $M_l = \sum_{k=1}^{K} M_l^k$. Also define $I_l^k(m)$ as the total number of failures observed prior to the mth failure during test case l occurring under operation k. Thus, prior to the mth failure observed under operation k during test case l, there will be $N_l - I_l^k(m)$ faults remaining in the software with the failure rate $(N_l - I_l^k(m))\lambda_{lk}$.

Given data $\mathcal{D} = \{U_l(1), \ldots, U_l(M_l), M_l^k, k = 1, \ldots, K, l = 1, \ldots, L\}$ observed from L test cases performed in sequence, using K operations, the likelihood function of the initial number of faults N and $\boldsymbol{\lambda} = (\lambda_{11}, \ldots, \lambda_{1K}, \ldots, \lambda_{L1}, \ldots, \lambda_{LK})$ is given by

$$L(N, \boldsymbol{\lambda}|\mathcal{D}) = \prod_{l=1}^{L} \frac{N_l!}{(N_l - M_l)!} e^{-(N_l - M_l)\lambda_l[\tau_l - U_l(M_l)]} \times \prod_{k=1}^{K} \lambda_{lk}^{M_l^k} e^{-\lambda_{lk}\{\sum_{m=1}^{M_l^k}[N_l - I_l^k(m)][U_l(I_l^k(m)+1) - U_l(I_l^k(m))]\}}. \tag{11}$$

Note that we can replace N_l with

$$N_l = N_{l-1} - M_{l-1} = N - \sum_{j=1}^{l-1} M_j \tag{12}$$

in the likelihood function.

As in the individual faults model, we assume independent gamma priors on each λ_{lk} with shape parameter a_{lk} and scale parameter b_{lk}. The prior for N is assumed to be Poisson with mean μ, and is independent of $\boldsymbol{\lambda}$. As before, a fully Bayesian analysis can be made using the Gibbs sampler. The implementation of the Gibbs sampler requires the full posterior conditionals $p(N|\boldsymbol{\lambda}, \mathcal{D})$ and $p(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D})$ for $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, K$.

The full conditionals $p(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D})$ can be obtained as independent gamma densities given by

$$(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D}) \sim \text{Gamma } (M_l^k + a_{lk}, \bar{b}_{lk}), \tag{13}$$

where $\bar{b}_{lk}$ and other details are given in the Appendix. In obtaining the full conditional of N, it can be shown that

$$(N - M|\boldsymbol{\lambda}, \mathcal{D}) \sim \text{Poisson } (\mu e^{-\sum_{l=1}^{L}\sum_{k=1}^{K}\lambda_{lk}g_{lk}(\mathcal{D})}), \tag{14}$$

where $M = \sum_{l=1}^{L} M_l$ and $g_{lk}(\mathcal{D})$ is given in the Appendix.

The above implies that as in the individual faults model, all the posterior distributions can be evaluated by recursively simulating from the full conditionals in a straightforward manner.

We note that in both the individual and common faults models the priors on initial number of faults are Poisson distributions with means assumed to be known; see (2) and (4). It is possible to treat the Poisson means as unknowns and describe uncertainty about them using a prior distribution. If the prior distributions are specified as independent gamma densities for μ_k's under the individual faults model and a gamma density under the common faults model, Bayesian inference can still be developed using the Gibbs sampler. In the following section details of such analysis will be presented using a special case.

5. A SPECIAL CASE

A interesting special case of the common faults model arises when each test case is associated with a single operation, that is, $L = K, \lambda_{lk} = 0$ for all $l \neq k$ and $\lambda_{lk} = \lambda_k$ when $l = k$. This implies that each test case is designed to test a given operation in the profile. We consider this special case for simplicity in our presentation only and not due to any significant difficulty in computation or implementation. It applies, in particular, in situations where test cases are designed to test for specific operations. The results obtained in the previous section are valid with the further simplification that $I_l^k(m) = 0$ for all $l \neq k$ and $I_k^k(m) = m - 1$. When $K = 1$, we obtain the Jelinski and Moranda [5] model for which Bayesian analysis has been done by Meinhold and Singpurwalla [8] and more recently by Kuo and Yang [7] by using a Gibbs sampler.

By using independent gamma priors with $\lambda_k \sim \text{Gamma } (a_k, b_k)$ and a Poisson prior for N, it can be shown that the full conditionals are given by

$$(\lambda_k|\lambda^{(-k)}, N, \mathcal{D}) \sim \text{Gamma } \left(M_k + a_k, b_k + \sum_{m=1}^{M_k} U_k(m) + \left(N - \sum_{j=1}^{k} M_j \right) \tau_k \right) \tag{15}$$

and

$$(N - M|\boldsymbol{\lambda}, \mathcal{D}) \sim \text{Poisson } (\mu e^{-\sum_{k=1}^{K}\lambda_k\tau_k}). \tag{16}$$

In specifying the prior distributions for λ_k's, the prior parameters (a_k, b_k) can be specified by eliciting a *best guess* value for λ_k and an uncertainty measure about the best guess from software engineers. The best guess can be thought as the mean, that is, a_k/b_k or the mode, that is, $(a_k-1)/b_k$, of the gamma density. The elicited uncertainty measure can be used to obtain a prior variance estimate, that is, a_k/b_k^2 for the gamma density. Based on these, the values of (a_k, b_k) can be specified by solving the mean and variance relationships. In specifying the prior mean of N, a similar approach can again be used or, alternatively, mean μ can be treated as an unknown quantity and a prior distribution can be specified for μ.

The case where μ is unknown can be handled easily. Again such an analysis was considered in Kuo and Yang [7] for the special case $K = 1$. We can assume a gamma prior for μ such that

$$\mu \sim \text{Gamma } (\alpha, \beta) \tag{17}$$

independent of $\boldsymbol{\lambda}$. Now the implementation of the Gibbs sampler requires the full conditionals $p(N|\boldsymbol{\lambda}, \mathcal{D}, \mu), p(\lambda_k|\lambda^{(-k)}, N, \mathcal{D}, \mu)$ for $k = 1, 2, \ldots, K$ and $p(\mu|N, \boldsymbol{\lambda}, \mathcal{D})$. The first two are specified as before by (16) and (15). It can be easily shown that

$$(\mu|N, \boldsymbol{\lambda}, \mathcal{D}) \sim \text{Gamma } (N + \alpha, \beta + 1). \tag{18}$$

Thus, the implementation of the Gibbs sampler is again straightforward.

Alternatively, we can use a dependent prior by assuming an ordering of the λ_k's using models introduced in Erkanlı, Mazzuchi, and Soyer [3] for analysis of the reliability growth process. The implementation of the Gibbs sampler in this case requires an additional rejection or a Metropolis step in each iteration. The details will be discussed in the sequel.

A reasonable assumption is that testing is done sequentially such that $\{\lambda_k\}$ is a nonincreasing sequence in k, that is,

$$+\infty > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K > 0. \tag{19}$$

In other words, testing is performed starting from those operations with a higher rate of failures.

In specifying the model for $\{\lambda_k\}$ following Erkanlı, Mazzuchi, and Soyer [3], we consider the transformation

$$R_k = e^{-\lambda_k} \tag{20}$$

and specify an ordered Dirichlet prior for $\{R_k\}$ as

$$p(R_1, \ldots, R_K, R_{K+1}) \propto \prod_{k=1}^{K+1} (R_k - R_{k-1})^{ba_k - 1}, \tag{21}$$

where $R_0 \equiv 0, R_{K+1} = 1$ and $b, a_k > 0$ for all k with $\sum_{k=1}^{K+1} a_k = 1$. The distribution is defined over the simplex $\{0 < R_1 < \cdots < R_K < 1\}$, and it can be shown that all marginal and conditional distributions are beta densities. For example, defining $a_k^* = \sum_{j=1}^{k} a_k$, it can be shown that the marginal distributions are given by

$$R_k \sim \text{Beta } (ba_k^*, b(1 - a_k^*)), \tag{22}$$

where $E[R_k] = a_k^*$ and b is a degree of belief parameter with high values of b reflecting high precision for the distribution. It can also be shown that the conditional distributions are given by

$$p(R_k|R^{(-k)}) = p(R_k|R_{k-1}, R_{k+1}) = \text{Beta}\ ((ba_k, ba_{k+1}); R_{k-1}, R_{k+1}), \tag{23}$$

which is a truncated beta density over (R_{k-1}, R_{k+1}).

Given the above prior, the λ_k's are defined over the simplex $+\infty > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K > 0$ as desired due to the one-to-one transformation. As in the previous case, with the independent priors on λ_k's, the posterior distributions can not be obtained analytically. Thus, we will again use a Gibbs sampler. In implementation of the Gibbs sampler we need the full conditionals $p(N|\lambda, \mathcal{D})$ and $p(\lambda_k|\lambda^{(-k)}, N, \mathcal{D})$ for $k = 1, 2, \ldots, K$. The full conditional of N will not change, so we will focus only on the full conditionals of λ_k's. In so doing, samples will be generated from the full conditionals $p(R_k|R^{(-k)}, N, \mathcal{D})$ and the transformation $R_k = e^{-\lambda_k}$ will be used to obtain the λ_k's. We now have

$$p(R_k|R^{(-k)}, N, \mathcal{D}) \propto L(R_k|R^{(-k)}, N, \mathcal{D}) p(R_k|R^{(-k)}), \tag{24}$$

where $L(R_k|R^{(-k)}, N, \mathcal{D})$ is obtained from

$$L(\lambda_k|\lambda^{(-k)}, N, \mathcal{D}) \propto \lambda_k^{M_k} e^{-\lambda_k[\sum_{m=1}^{M_k} U_k(m) + (N_k - M_k)\tau_k]} \tag{25}$$

using $R_k = e^{-\lambda_k}$. It is important to note that the conditional likelihood $L(R_k|R^{(-k)}, N, \mathcal{D})$ is constrained over (R_{k-1}, R_{k+1}).

The exact form of $p(R_k|R^{(-k)}, N, \mathcal{D})$ is not known and thus a standard rejection sampling method is employed within the Gibbs sampler as follows:

STEP 1: Generate R_k from the prior conditional density $p(R_k|R^{(-k)}) = p(R_k|R_{k-1}, R_{k+1})$, which is a truncated beta density as in (23).

STEP 2: Generate independently a 0–1 uniform random variate u.

STEP 3: Compute the ratio

$$\frac{L(R_k|R^{(-k)}, N, \mathcal{D})}{L(\hat{R}_k|R^{(-k)}, N, \mathcal{D})}, \tag{26}$$

where $\hat{R}_k$ is the maximum of the conditional likelihood and is available analytically as

$$\hat{R}_k = \begin{cases} R_{k-1}, & \text{if } e^{-M_k/T_k(N)} \leq R_{k-1}, \\ e^{-M_k/T_k(N)}, & \text{if } R_{k-1} < e^{-M_k/T_k(N)} < R_{k+1}, \\ R_{k+1}, & \text{if } R_{k+1} \leq e^{-M_k/T_k(N)}, \end{cases} \tag{27}$$

where $T_k(N) = \sum_{m=1}^{M_k} U_k(m) + (N - \sum_{j=1}^{k} M_j)\tau_k$.

STEP 4: Accept R_k if

$$u \leq \frac{L(R_k|R^{(-k)}, N, \mathcal{D})}{L(\hat{R}_k|R^{(-k)}, N, \mathcal{D})}; \tag{28}$$

otherwise reject R_k, go to step 1, and repeat the process. This way we can implement the Gibbs sampler as before.

6. OPTIMAL TESTING STRATEGY

Testing the software to remove the faults before release is an important decision problem which leads to interesting optimization formulations. In many cases, one has to determine how long to test the software. On the one hand, we want to test the software as extensively as possible to increase its reliability by removing the faults, while, on the other hand, we want to release it as soon as possible without incurring excessive testing costs. The problem may be further complicated by due dates for release as well as minimum reliability requirements. There is an apparent tradeoff between the reliability of the software and testing cost. The main objective in the software release problem is the development of stopping rules for the testing procedure that will minimize the expected total cost or maximize the expected total benefit. There is a vast amount of literature on the subject as referenced in the survey by Özekici and Çatkan [11]. The reader is referred to Forman and Singpurwalla [4], Koch and Kubat [6], Yamada and Osaki [20], Ross [15], Bai and Yun [1], and Brown, Maghsoodloo, and Deason [2] for some examples.

An optimal testing problem involving the operational profile is first introduced by Özekici, Altınel, and Özçelikyürek [13]. Their model is similar to ours in the sense that the software is tested in a prescribed deterministic order of test cases, where the failure time distribution for each operation in each test case is exponential. However, they discuss a static decision problem with a non-Bayesian framework. All failure rates $\{\lambda_k\}$, as well as the expected number of initial faults, are assumed to be known. The testing durations $\{\tau_k\}$ are determined at time zero, and they remain unchanged until the end. The objective is to minimize the expected total cost where all cost parameters depend on the specific operation performed.

In what follows we are considering the special case of the common faults model discussed in Section 3. In this setup, software testing is done in a sequential manner with a prescribed duration of testing τ_k for operation k. Thus, choice of τ_k for each operation $k, k = 1, \ldots, K$, constitutes the testing strategy. The initial testing strategy is determined prior to any testing, and, as testing is done in a sequential manner and failure data are observed, the optimal strategy is revised for the remaining test effort. In other words, if testing has been completed for the first n operations and failure data $\mathcal{D}^n = \{(U_k(1), \ldots, U_k(M_k), k = 1, \ldots, n\}$ are observed, uncertainty about the unknown parameters as well as the optimal test durations τ_k's for environments $k = n+1, \ldots, K$, is revised based on the test data $\mathcal{D}^n$. Note that $\mathcal{D}^0$ denotes the information prior to any testing.

As in most models in the literature, there are three cost factors. During operation k, the testing cost per unit time is c_k, the cost of debugging a fault before release is d_k while this cost is e_k after release. The operational profile is represented by the distribution $\{p_k\}$, where p_k is the proportion of time that the software will perform operation k after release. We also suppose that this is also the probability that a remaining fault will cause a failure of operation k after release. It is shown by Özekici, Altınel, and Özçelikyürek [13] that the static decision problem is in fact a nonlinear programming problem that can be stated as

$$\min_{0 \le \tau_k < +\infty} \sum_{k=1}^{K} \left[c_k \tau_k - E[N] f_k (1 - e^{-\lambda_k \tau_k}) \prod_{j=1}^{k-1} e^{-\lambda_j \tau_j} \right] \tag{29}$$

where $f_k = \sum_{j=1}^{K} p_j e_j - d_k$ is the benefit obtained by removing a fault during testing in operation k before it is released. They also prove that (29) has an explicit solution under reasonable assumptions.

We consider the case of independent prior for $\boldsymbol{\lambda}$ as in Section 3. Assume that testing is completed for the first $n-1$ operations and $\mathcal{D}^{n-1}$ has been observed. Following (29), the optimal expected

cost associated with testing for operations $k = n, \ldots, K$ is now given by

$$\min_{\substack{0\le\tau_k<+\infty\\ k=n,\ldots,K}} \sum_{k=n}^{K} \left[c_k\tau_k - E[N_n|\mathcal{D}^{n-1}] f_k(1 - L_k^n(\tau_k)) \prod_{j=n}^{k-1} L_j^n(\tau_j) \right], \tag{30}$$

where the decision variables are $\tau_n, \ldots, \tau_K$ and

$$L_j^n(\tau_j) = E[e^{-\lambda_j\tau_j}|\mathcal{D}^{n-1}]. \tag{31}$$

Note that (30) reduces to (29) if $n = 1$ so that we are at the beginning of testing and all of the parameters are known with certainty. This implies $E[N_1|\mathcal{D}^0] = E[N] = \mu$ and $L_j^1(\tau_j) = E[e^{-\lambda_j\tau_j}|\mathcal{D}^0] = e^{-\lambda_j\tau_j}$.

The above form (30) follows from the prior independence of the λ_j's for $j = 1, \ldots, K$. Under this prior, it can be shown that given $\mathcal{D}^{n-1}$

$$p(\lambda, N|\mathcal{D}^{n-1}) \propto p(N) \left[\prod_{k=1}^{n-1} \lambda_k^{M_k+a_k-1} e^{-\lambda_k[b_k+\sum_{j=1}^{M_k} U_k(j)+(N_k-M_k)\tau_k]} \right] \prod_{j=n}^{K} \lambda_j^{a_j-1} e^{-\lambda_j b_j}, \tag{32}$$

implying that $\lambda_n, \ldots, \lambda_K$ are independent and not revised from the prior, that is,

$$(\lambda_j|N, \mathcal{D}^{n-1}) \sim \text{Gamma}\,(a_j, b_j) \tag{33}$$

for $j = n, \ldots, K$. Thus, it follows from the above that

$$L_j^n(\tau_j) = E[e^{-\lambda_j\tau_j}|\mathcal{D}^{n-1}] = \left(\frac{b_j}{b_j + \tau_j}\right)^{a_j} \tag{34}$$

and $E[N_n|\mathcal{D}^{n-1}]$ can be evaluated from the Gibbs sampler introduced in Section 4 by replacing $\mathcal{D}$ with $\mathcal{D}^{n-1}$.

Using (34) and following the same transformation as in Özekici, Altınel, and Özçelikyürek [13], the optimization problem (30) can be restated as

$$\min_{\substack{0\le\tau_k<+\infty\\ k=n,\ldots,K}} \sum_{k=n}^{K} \left[c_k\tau_k + E[N_n|\mathcal{D}^{n-1}](f_k - f_{k+1}) \prod_{j=n}^{k} \left(\frac{b_j}{b_j + \tau_j}\right)^{a_j} \right] \tag{35}$$

with $f_{K+1} = 0$. This representation provides computational advantage since the objective function in (35) is convex if $d_1 \le d_2 \le \cdots \le d_K$. This is a desired property in optimization in which case first-order optimality conditions are sufficient to claim global optimality. Convexity follows since the first function inside the summation in (35) is linear, thus convex, $f_1 \ge f_2 \ge \cdots \ge f_{K+1}$, and (34) is a nonnegative, convex decreasing function. In the static framework with known parameters where $L_j^n(\tau_j) = e^{-\lambda_j\tau_j}$. Özekici, Altınel, and Özçelikyürek [13] use convexity arguments to identify an explicit optimal solution.

In this Bayesian case, the new form of $L_j^n(\tau_j)$ in (34) does not necessarily give an explicit solution. In the static non-Bayesian framework, Özekici, Altınel, and Angün [14] discuss in more detail optimization problems with the structure given by (35). They present optimality conditions that must be satisfied by the optimal solution. These conditions suggest a recursive algorithm that can be used to identify the optimal solution. The computational issues do not really impose significant difficulties since convexity still holds and we can use an optimization package to identify the optimal strategy. It is clear that the solution depends on the model parameters. In the static case with exponential failures, the explicit solution given by (27) in Özekici, Altınel, and Özçelikyürek [13] indicate that the optimal test durations are quite sensitive to changes in the model parameters since they involve logarithmic functions. In our setting, however, we cannot express this sensitivity explicitly since (35) does not have an explicit solution. But the results on the static case and the numerical solutions of some examples we analyzed indicate that the optimal testing durations can be highly sensitive to changes in the parameters.

Using the above setup, we can determine the optimal testing strategy in a dynamic manner. Prior to any testing, an optimal strategy is determined for all operations $k = 1, \ldots, K$ by minimizing (30) using the prior information on N and $\boldsymbol{\lambda}$. Thus, the optimal test durations $(\tau_1^{(0)}, \ldots, \tau_K^{(0)})$ are specified and the software is tested under operation 1 for a duration of $\tau_1^{(0)}$. Once the failure data $\mathcal{D}^1$ become available, uncertainty about N and $\boldsymbol{\lambda}$ are revised using the Bayesian setup presented in Section 4 and the testing strategy for remaining operations $k = 1, \ldots, K$ are updated from $(\tau_2^{(0)}, \ldots, \tau_K^{(0)})$ to $(\tau_2^{(1)}, \ldots, \tau_K^{(1)})$. This process of testing and revising the uncertainties and testing continue until the software is tested for all operations.

7. ILLUSTRATION

Musa [9] presents a strong case for using the operational profile in software reliability engineering. He asserts that the benefit-to-cost ratio in developing and applying the operational profile is typically 10 or more. This involves not only testing the software, but the development stage as well. He outlines a five-step procedure as: (1) identification of the customer profile, (2) establishing the user profile, (3) defining the system-mode profile, (4) determination of the functional profile, and (5) determination of the operational profile itself. These five steps are explained in detail by Musa [9] for a number of applications.

Table 1. Operational profile for billing system.

Operation	Occurrence probability
Residential, no calling plan, paid	0.5940
Residential, national calling plan, paid	0.1580
Business, no calling plan, paid	0.1485
Business, national calling plan, paid	0.0396
Residential, international calling plan, paid	0.0396
Business, international calling plan, paid	0.0099
Residential, no calling plan, delinquent	0.0060
Residential, national calling plan, delinquent	0.0016
Business, no calling plan, delinquent	0.0015
Business, national calling plan, delinquent	0.0004
Residential, international calling plan, delinquent	0.0004
Business, international calling plan, delinquent	0.0001

Table 2. Model parameters.

k	a_k	b_k	c_k	f_k
1	1	10	1	50
2	1	15	2	80
3	1	20	4	100

One of the applications, for example, concerns a data-driven telephone billing system where the operations are classified by 2 types of service (residential or business), usage of 3 discount calling plans (none, national or international) and 2 types of payment status (paid or delinquent). The profile thus results in $K = 2 \times 3 \times 2 = 12$ operations with occurrence probabilities given in Table 1.

Table 3. The optimal testing strategy.

n	$\tau_1^{(n-1)}$	$\tau_2^{(n-1)}$	$\tau_3^{(n-1)}$	$E[N\|\mathcal{D}^{n-1}]$	$E[N_n\|\mathcal{D}^{n-1}]$	M_n
1	207.93	104.63	0	100	100	41
2	—	77.53	0	98.08	57.08	32
3	—	—	0	88.81	15.81	—

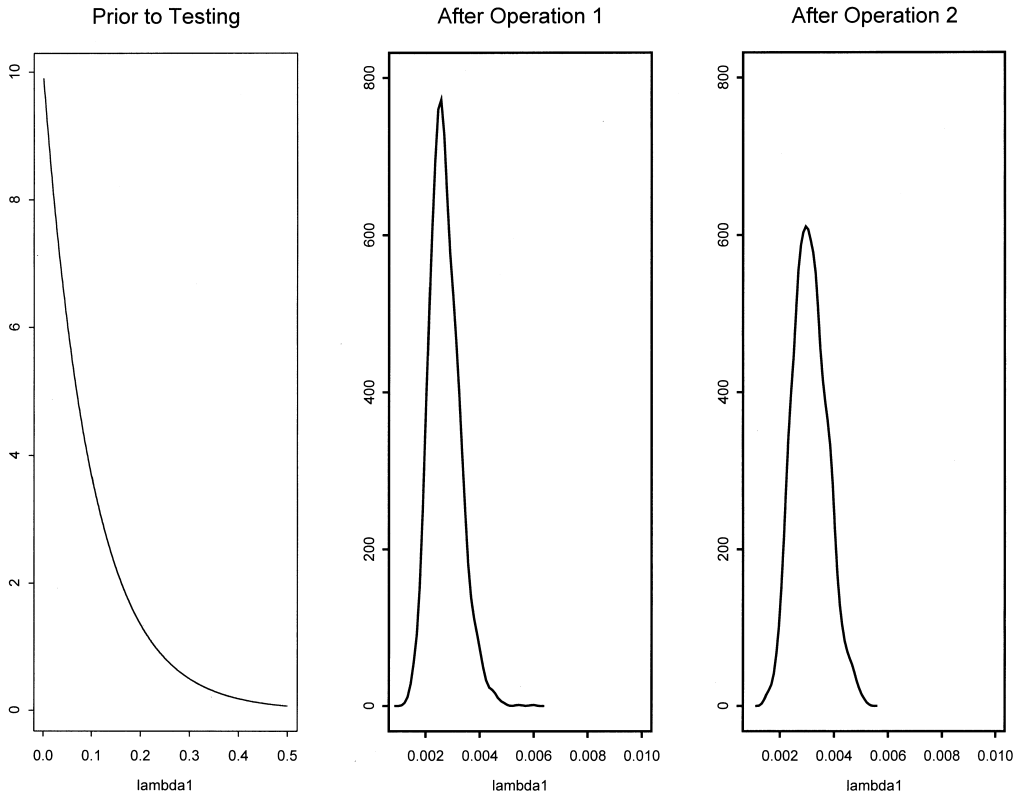

Figure 1. Prior and posterior distributions of λ_1.

While 59.4% of billing operations are on (residential, no calling plan, paid) type accounts, only 0.01% of operations involve (business, international calling plan, delinquent) type accounts. Based on these facts on the profile of operations, Musa [9] suggests that operations in testing should be selected in accordance with their occurrence probabilities. Furthermore, he states that operations can also be classified with respect to criticality: value added (increased revenue or reduced cost) or the severity of the effect when they fail. In this case, the weighted occurrence probabilities $p'_k = c_k p_k / \sum_l c_l p_l$ can be used to drive the testing plan, where c_k stands for the criticality of operation k.

Our approach on the testing problem is based on economic analysis that aims to minimize the expected total cost. Therefore, the criticality factor involving the costs and the operational profile enter our nonlinear programming formulation in an indirect manner. It goes beyond a criticality ordering of the operations in choosing a test plan. However, the complete cost and failure data required by the model is not available. But, to illustrate our procedure, suppose that we choose the first $K = 3$ more frequent operations in our testing and use the arbitrary data set given in Table 2, where the prior distributions on the failure rates are taken to be exponential. Therefore, the shape parameters are $a_k = 1$ with varying values of the scale parameters. Both testing costs c_k and benefits f_k are taken to be in increasing order. Note if this is not true and there is an operation that is more costly to test but less beneficial, then that operation will never be tested.

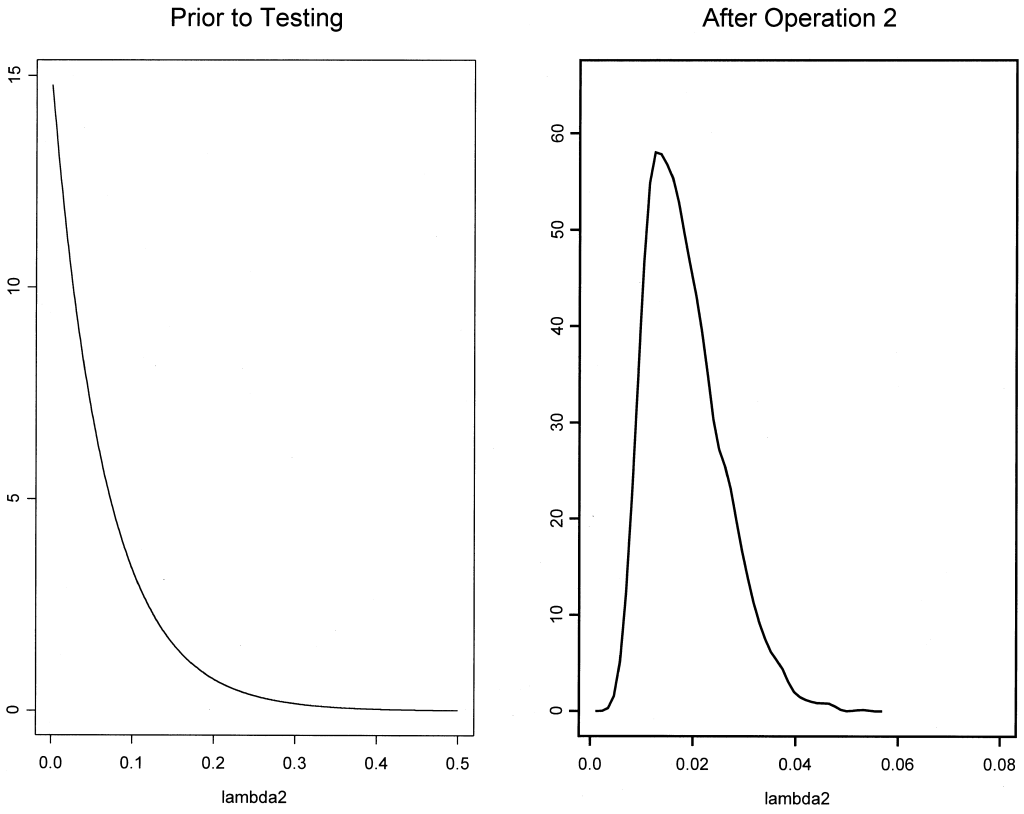

Figure 2. Prior and posterior distributions of λ_2.

The expected number of initial faults is taken arbitrarily to be 100. Solving the optimization problem (35) sequentially for $n = 1, 2, 3$ we obtain the testing policy and results summarized in Table 3. The failure times during each operation are generated through simulation because of unavailability of failure data in testing.

The initial policy is to test the software using operation 1 for $\tau_1^{(0)} = 207.93$ units of time. If this was a static problem with no updating of parameters using information gathered, then operation 2 would be tested for $\tau_2^{(0)} = 104.63$ units of time and it would not be economically viable to test operation 3 since $\tau_3^{(0)} = 0$. The prior information yields an expected number of 100 initial faults. During the testing of operation 1, simulation resulted in $M_1 = 41$ failures at different times over [0, 207.93]. The posterior analysis updated the initial number of faults to $E[N|\mathcal{D}^1] = 98.08$ and the expected number of faults remaining reduces to $E[N_2|\mathcal{D}^1] = 98.08 - 41 = 57.08$. Using this fact, the new optimization problem now gives $\tau_2^{(1)} = 77.53$ and $\tau_3^{(1)} = 0$ so that optimal test duration of operation 2 is decreased substantially while it is still not optimal to test operation 3. Simulation now resulted in $M_2 = 32$ failures at various times during [207.93, 285.46]. This leads to the posterior mean $E[N|\mathcal{D}^2] = 88.81$ and the expected number of faults remaining becomes $E[N_3|\mathcal{D}^2] = 88.81 - 41 - 32 = 15.81$. At the last stage, we find $\tau_3^{(2)} = 0$ and no testing should be done for operation 3.

At each stage, the posterior distributions of the failure rates λ_1 and λ_2 are computed as in Figure 1 and 2, respectively. Note that the prior distributions are exponential since $a_1 = a_2 = 1$.

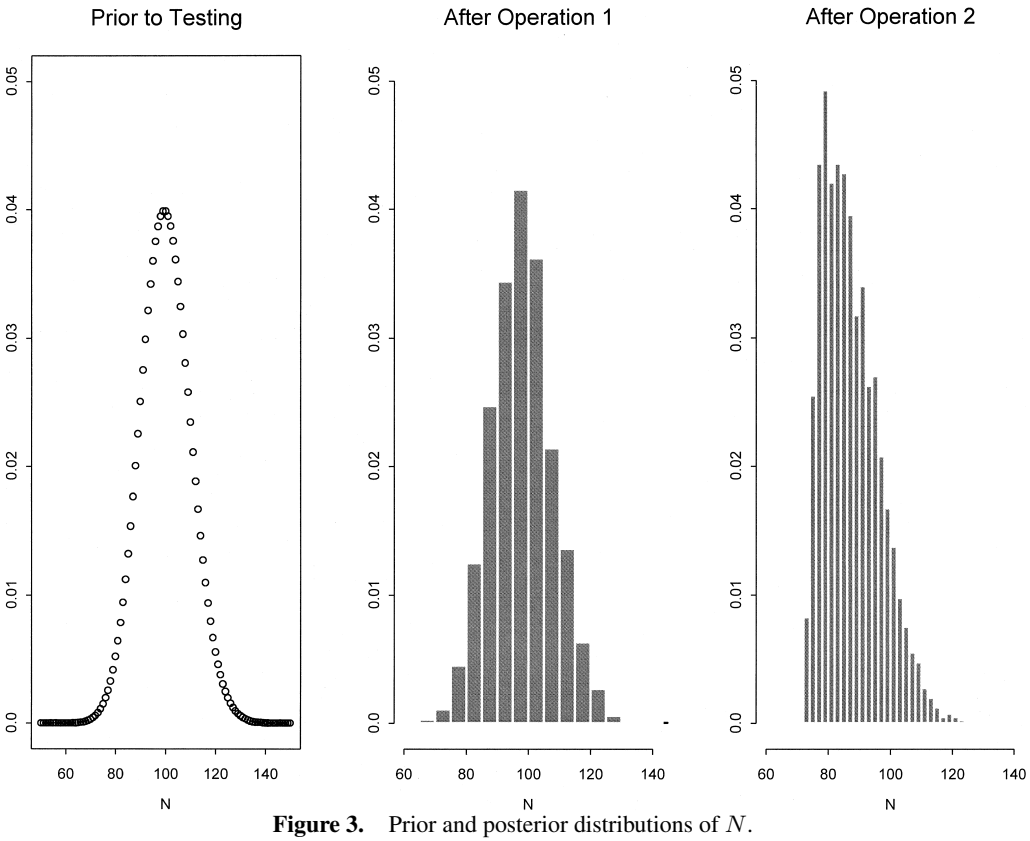

Figure 3. Prior and posterior distributions of N.

However, the variances of the distributions decrease as we update them using the information gathered during tests. The changes in the distribution of the number of faults N is reflected in Figure 3. The initial distribution is Poisson with mean 100 which decreases to 98.08 after the first test and to 88.81 at the end.

APPENDIX

A.1. Gibbs Sampler for the Individual Faults Model

In obtaining the full conditional distribution of λ_{lk}, the conditional likelihoods can easily be obtained from (5). Given $\mathbf{N}$ and $\lambda^{(-lk)}$, the conditional likelihood of λ_{lk} is given by

$$L(\lambda_{lk}|\lambda^{(-lk)},\mathbf{N},\mathcal{D}) \propto \lambda_{lk}^{M_l^k} e^{-\lambda_{lk}\{\sum_{m=1}^{M_l} N_l^k(m)[U_l(m)-U_l(m-1)]+(N^k-\sum_{j=1}^{l} M_j^k)[\tau_l-U_l(M_l)]\}}, \tag{36}$$

and the full conditionals $p(\lambda_{lk}|\lambda^{(-lk)},\mathbf{N},\mathcal{D})$ are

$$p(\lambda_{lk}|\lambda^{(-lk)},\mathbf{N},\mathcal{D}) \propto \lambda_{lk}^{M_l^k+a_{lk}-1} e^{-\lambda_{lk}b_{lk}} \tag{37}$$

where

$$\bar{b}_{lk} = b_{lk} + \sum_{m=1}^{M_l} N_l^k(m)[U_l(m)-U_l(m-1)] + \left(N^k - \sum_{j=1}^{l} M_j^k\right)[\tau_l - U_l(M_l)]. \tag{38}$$

Thus, the full conditionals

$$(\lambda_{lk}|\lambda^{(-lk)},\mathbf{N},\mathcal{D}) \sim \text{Gamma } (M_l^k + a_{lk}, \bar{b}_{lk}), \tag{39}$$

which are independent gamma densities for $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, K$.

For the full conditional distributions $p(N^k N^{(-k)}, \boldsymbol{\lambda}, \mathcal{D})$, we can write them as proportional to

$$\prod_{l=1}^{L} \frac{N_l^k!}{(N_l^k - M_l^k)!} e^{-\lambda_{lk}\{\sum_{m=1}^{M_l} N_l^k(m)[U_l(m)-U_l(m-1)]+(N^k-\sum_{j=1}^{l} M_j^k)[\tau_l-U_l(M_l)]\}} \frac{e^{-\mu_k}(\mu_k)^{N_1^k}}{N_1^k!}.$$

Using the fact that $N^k = N_1^k$, $N_l^k(m) = N^k - \sum_{j=1}^{l-1} M_j^k - M_l^k(m)$, $N_l^k = N_{l-1}^k - M_{l-1}^k$, and

$$\prod_{l=1}^{L} \frac{N_l^k!}{(N_l^k - M_l^k)!} \frac{1}{N_1^k!} = \frac{1}{(N_L^k - M_L^k)!} = \frac{1}{(N_1^k - \sum_{l=1}^{L} M_l^k)!},$$

we can write the full conditional distribution of N^k after some algebra as

$$p(N^k|N^{(-k)},\boldsymbol{\lambda},\mathcal{D}) \propto \frac{1}{(N^k - M^k)!} (\mu_k e^{-\sum_{l=1}^{L} \lambda_{lk}\tau_l})^{N^k} \tag{40}$$

where $M^k = \sum_{l=1}^{L} M_l^k$. We note that given $(\lambda_{1k}, \ldots, \lambda_{Lk})$ and the data, N^k is independent of $N^{(-k)}$ and the above implies that

$$(N^k - M^k | N^{(-k)}, \boldsymbol{\lambda}, \mathcal{D}) \sim \text{Poisson } (\mu_k e^{-\sum_{l=1}^{l} \lambda_{lk}\tau_l}). \tag{41}$$

A.2. Gibbs Sampler for the Common Faults Model

The full conditional distributions, $p(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D})$ can be obtained as

$$p(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D}) \propto \lambda_{lk}^{M_l^k + a_{lk} - 1} e^{-\lambda_{lk}\bar{b}_{lk}}, \tag{42}$$

where

$$\bar{b}_{lk} = b_{lk} + (N_l - M_l)[\tau_l - U_l(M_l)] + \sum_{m=1}^{M_l^k} [N_l - I_l^k(m)][U_l(I_l^k(m)+1) - U_l(I_l^k(m))], \tag{43}$$

implying that

$$(\lambda_{lk}|\lambda^{(-lk)}, N, \mathcal{D}) \sim \text{Gamma } (M_l^k + a_{lk}, \bar{b}_{lk}) \tag{44}$$

and λ_{lk} is independent of $\lambda^{(-lk)}$.

For the full conditional of N, we first recognize that

$$\prod_{l=1}^{L} \frac{N_l!}{(N_l - M_l)!} \frac{1}{N_1!} = \frac{1}{(N_1 - \sum_{l=1}^{L} M_l)!}$$

and using the fact that $N = N_1$ and $N_l = N_{l-1} - M_{l-1} = N - \sum_{j=1}^{l-1} M_j$, we can show that

$$p(N|\boldsymbol{\lambda}, \mathcal{D}) \propto \frac{1}{(N-M)!} (\mu e^{-\sum_{l=1}^{L}\sum_{k=1}^{K} \lambda_{lk} g_{lk}(\mathcal{D})})^N, \tag{45}$$

where $M = \sum_{l=1}^{L} M_l$ and

$$g_{lk}(\mathcal{D}) = [\tau_l - U_l(M_l)] + \sum_{m=1}^{M_l^k} [N_l - I_l^k(m)][U_l(I_l^k(m)+1) - U_l(I_l^k(m))]. \tag{46}$$

The above implies that

$$(N - M|\boldsymbol{\lambda}, \mathcal{D}) \sim \text{Poisson } (\mu e^{-\sum_{l=1}^{L}\sum_{k=1}^{K} \lambda_{lk} g_{lk}(\mathcal{D})}) \tag{47}$$

and, as in the individual faults model, all the posterior distributions can be evaluated by recursively simulating from the full conditionals in a straightforward manner.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation Grant INT-9602081 and Boğaziçi University Research Fund Grant 97HA303.

REFERENCES

[1] D.S. Bai and W.Y. Yun, Optimum number of errors corrected before releasing a software system, IEEE Trans Reliab 37 (1988), 41–44.

[2] D.B. Brown, S. Maghsoodloo, and W.H. Deason, A cost model for determining the optimal number of software test cases, IEEE Trans Software Eng 15 (1989), 218–221.

[3] A. Erkanlı, T.A. Mazzuchi, and R. Soyer, Bayesian computations for a class of reliability growth models, Technometrics 40 (1998), 14–23.

[4] 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.

[5] Z. Jelinski and P. Moranda, "Software reliability research," Statistical computer performance evaluation, W. Freiberg, Editor, Academic, New York, 1972, pp. 465–484.

[6] H.S. Koch and P. Kubat, Optimal release time of computer software, IEEE Trans Software Eng 9 (1983), 323–327.

[7] L. Kuo and T.Y. Yang, Bayesian computations of software reliability, J Comput Graphical Stat 4 (1995), 65–82.

[8] R.J. Meinhold and N.D. Singpurwalla, Bayesian analysis of a commonly used model for describing failure times, Statistician 32 (1983), 168–173.

[9] J.D. Musa, Operational profiles in software reliability engineering, IEEE Software 10 (1993), 14–32.

[10] J.D. Musa, "The operational profile," Reliability and maintenance of complex systems, S. Özekici, Editor, NATO ASI Series Volume F154, Springer-Verlag, Berlin, 1996, pp. 332–343.

[11] S. Özekici and N.A. Çatkan, Software release models: A survey, Proceedings of the 5th International Symposium on Computer and Information Sciences (ISCIS V), A.E. Harmancıand E. Gelenbe, Editors, 1992, pp. 203–210.

[12] S. Özekici and R. Soyer, Reliability of software with an operational profile, working paper, The George Washington University, Department of Management Science, Washington, DC, 2000.

[13] S. Özekici, İ.K. Altınel, and S. Özçelikyürek, Testing of software with an operational profile, Nav Res Logistics 47 (2000), 620–634.

[14] S. Özekici, İ.K. Altınel, and E. Angün, A general software testing model involving operational profiles, Probab Engineering Inf Sci 15 (2001), 519–533.

[15] S.M. Ross, Software reliability: The stopping rule problem, IEEE Trans Software Eng 11 (1985), 1472–1475.

[16] N.D. Singpurwalla, T.A. Mazzuchi, S. Özekici, and R. Soyer, Software reliability and the operational profile, technical report, GWU/IRRA Series TR-99/1, The George Washington University, Washington, DC, 1999.

[17] J.A. Whittaker and J.H. Poore, Markov analysis of software specifications, ACM Trans Software Eng Methodol 2 (1993), 93–106.

[18] J.A. Whittaker and M.G. Thomason, A Markov chain model for statistical software testing, IEEE Trans Software Eng 20 (1994), 812–824.

[19] C. Wohlin and P. Runeson, Certification of software components, IEEE Trans Software Eng 20 (1994), 494–499.

[20] S. Yamada and S. Osaki, Cost-reliability optimal release policies for software systems, IEEE Trans Reliab 34 (1985), 422–424.