Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 149 (2003) 459–474
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Stochastics and Statistics

Reliability of software with an operational profile

S. Özekici [a,*], R. Soyer [b]

[a] Department of Industrial Engineering, Koç University, 80910 Sarıyer-İstanbul, Turkey
[b] Department of Management Science, The George Washington University, Washington, DC 20052, USA

Received 9 October 2000; accepted 13 May 2002

Abstract

This article provides the stochastic and statistical framework to model software reliability in the presence of an operational profile. The software system is used under a randomly changing operational process so that the failure characteristics depend on the specific operation performed. The operational process describes, in a probabilistic sense, how the software is utilized by the users. The time to failure distribution for each fault is exponentially distributed with a rate that depends on the operation. As soon as a failure is experienced, the fault that caused the fault is removed immediately with certainty. We discuss several issues related to software reliability and statistical inference.

Keywords: Reliability; Stochastic processes; Operational profile; Markovian analysis; Bayesian analysis

1. Introduction

An undesirable restriction in almost all of the software reliability models in the literature is that the parameters of the software failure process are independent of the operation that the software performs. This implicitly assumes that the software is used for a single operation only, or that there are no significant differences in the model parameters for the variety of operations that the system is designed to perform. To overcome this restriction, the concept of operational profiles was introduced by Musa et al. [10] as an important part of the technology and practice of software reliability engineering. Musa [8,9] describes how operational profiles can be built and states that there is substantial benefit to be gained by applying it. It can increase user satisfaction by capturing their needs more precisely, satisfy important user needs faster, reduce costs with reduced operation software, speed up the development and improve productivity by allocating resources in relation to use and criticality, reduce the system risk by more realistic testing, and make testing faster and more efficient.

Musa [8] 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

* Corresponding author. Tel.: +90-212-338-1723; fax: +90-212-338-1548.
E-mail addresses: sozekici@ku.edu.tr (S. Özekici), soyer@gwu.edu (R. Soyer).

0377-2217/03/$ - see front matter
doi:10.1016/S0377-2217(02)00461-7

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 [8] for a number of applications.

One of the applications, for example, concerns a data-driven telephone billing system where the operations are classified by two types of service (residential or business), usage of three discount calling plans (none, national or international) and two 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.

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 [8] 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. One of the main implications of the profile is that it directly affects software reliability since failure probabilities may depend on the specific operation that will be performed. Thus, the sequence of operations as well as their durations play a critical role in assessing software reliability.

An operational profile simply consists of the set of all operations that a system is designed to perform and their probabilities of occurrence. It provides a quantitative characterization of how the system will be used in the field, making it an essential ingredient of software reliability engineering. Technically speaking, one can think of an operational profile as a generic random variable that indicates the operations that will be performed by the system. Of course, it is only reasonable to assume that the failure processes generated by the system faults depend on the specific operations performed by the system. This fact is often overlooked with simplifying assumptions in reliability calculation and testing. The primary objective of this article is to propose a model on software reliability that incorporates the operational profile and discuss how probabilistic and Bayesian analysis can be performed on various quantities of interest. To our knowledge, this issue is not properly addressed in the software reliability engineering literature.

In a recent article, Özekici et al. [12] discuss the optimal testing of software with an operational profile. The software is tested in all operations sequentially in a controlled manner before it is released and the optimal policy is determined through nonlinear programming formulations. In the present setting, however, the operations are performed by the user in a random manner and our objective is to determine software reliability and other performance measures. Wohlin and Runeson [17] 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 [15] and Whittaker and Thomason [16]. In their approach, the sequence of "inputs" provided by the user is modelled as a Markov chain. This results in a model at the micro level

Table 1
Operational profile for billing system

Operation	Occurence probability
Residential, no calling plan, paid	0.5940
Residential, natinonal calling plan, paid	0.1580
Business, no calling plan, paid	0.1485
Business, natinonal 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

involving all possible values of input variables with a huge state 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. For a complete discussion on the stochastic foundations for describing and incorporating the operational profile, the reader is referred to Singpurwalla et al. [13].

We will formalize the concept of an operational profile and process mathematically in Section 2 and our software reliability model will be introduced in Section 3. This model is in fact computationally tractable as we demonstrate in Section 4 how various quantities involving the failure time and time of perfection can be computed. Statistical issues are addressed in Section 5 where we perform Bayesian analysis using the Gibbs sampler. Finally, we will illustrate our results by a numerical example in Section 6.

2. The operational process

An operation is an externally initiated task performed by a system "as built". A software system is usually designed to perform a set of well-defined operations or tasks. Suppose that the discrete set E denotes the set of all operations and that Y_t is the operation performed by the system at time t. We would therefore like to think of the continuous-time process $Y = \{Y_t;\ t \geqslant 0\}$ as the *operational process* and the state space E as the *operational space*.

If the stochastic process Y is ergodic with some limiting (stationary) distribution $\pi(i) = \lim_{t\to\infty} P[Y_t = i]$, then $\pi(i)$ is simply the proportion of time that operation i is performed in the long run. Thus, technically speaking, the pair (E, π) is the *operational profile* that consists of the set of all possible operations and their occurrence probabilities, as defined by Musa [8].

The analysis of the software failure process obviously depend on the stochastic structure of the operational process. In this exposition, we assume that Y is a Markov process. Briefly, this means that the sequence of operations performed is a Markov chain and the amount of time spent on each operation is exponentially distributed. More precisely, we let X_n denote the nth operation that the system performs and S_n be the time at which the nth operation starts. It is well-known that X is a Markov chain with some transition matrix

$$P(i,j) = P[X_{n+1} = j | X_n = i] \tag{1}$$

and

$$P[S_{n+1} - S_n > t | X_n = i] = \mathrm{e}^{-\mu(i)t} \tag{2}$$

so that the duration of the nth operation is exponentially distributed with rate $\mu(i)$ if this operation is i.

The relationship between the three processes Y, X and S is such that

$$Y_t = X_n \quad \text{on} \quad S_n \leqslant t < S_{n+1}. \tag{3}$$

In our analysis, we hope to make use of the well-known results on Markov processes to study the transient and ergodic behavior of the software system. Of course, our major concern will be software reliability or the probability of failure free operation during a specified time interval, and the number of failures encountered in time. The generator matrix A of Y is defined as $A(i,j) = \mu(i)(P(i,j) - I(i,j))$ where I is the identity matrix. It is known that the transition function $P_{ij}(t) = P[Y_t = j | Y_0 = i]$ has the matrix geometric form

$$P(t) = \mathrm{e}^{At} = \sum_{k=0}^{+\infty} \frac{t^k}{k!} A^k. \tag{4}$$

Since Y is a Markov process with a finite state space E, we can easily compute many quantities of interest using linear algebra. The reader is referred to any standard textbook on stochastic processes, for example Çınlar [1], for computational details. Assuming that E is irreducible, the profile π is the unique solution of the system of linear equations

$$\pi A = 0, \quad \sum_{i\in E} \pi(i) = 1. \tag{5}$$

If v is the limiting distribution of X such that $v(i) = \lim_{n\to\infty} P[X_n = i]$, then it is the unique solution of

$$v = vP, \quad \sum_{i\in E} v(i) = 1. \tag{6}$$

Moreover,

$$\pi(i) = \frac{v(i)/\mu(i)}{\sum_{j\in E}[v(j)/\mu(j)]} \tag{7}$$

for any operation i. Given that the initial operation is i, the expected number of operations performed until operation j is performed for the first time satisfies the system of linear equations

$$r(i,j) = 1 + \sum_{k\neq j} P(i,k) r(k,j). \tag{8}$$

The expected number of operations performed in between two performances of operation i is known to be

$$r(i,i) = \frac{1}{v(i)} \tag{9}$$

and, in more generality, the ratio $v(j)/v(i)$ gives the expected number of operation j performed in between two performances of operation i.

Considering the durations of the operations as well, the expected first passage time $m(i,j)$ from operation i to j satisfies

$$m(i,j) = \frac{1}{\mu(i)} + \sum_{k\neq j} P(i,k) m(k,j) \tag{10}$$

and the mean recurrence time of operation i is

$$m(i,i) = \frac{1}{\pi(i)\mu(i)}. \tag{11}$$

More precisely, $m(i,j)$ gives the expected future time at which operation j will be initiated if the software is performing operation i now.

Available models of software reliability do not provide a good match that fits the requirements of the operational profile. Simply stated, we need models where software failures and reliability depend very much on the operations that the software performs. The stochastic as well as deterministic model parameters are not necessarily constants as in almost all of the models, but they depend on the specific operations that the software performs. In a far more general context, this issues is addressed by Özekici [11] with various applications in inventory, queueing and reliability models. He proposes to use a so-called *environmental process* that modulates the parameters of the specific model. In an inventory model, for example, this could be any stochastic process that reflects the prevailing economic conditions and state of the economy that affects the parameters of the demand distribution and all cost factors. It affects the arrival and service rates in the queueing context. The use of the environmental process in reliability applications was originally introduced by Çınlar and Özekici [2] to measure the age of a device intrinsically and model stochastic dependence among components of a system.

There is an apparent similarity between the *operational process* in software engineering and the *environmental process*. They are both stochastic processes that affect, in one way or another, the stochastic as well as deterministic parameters of the model. Our methodology will be based on this similarity as we explain some specific details next.

3. Software failure process

An overview of software failure models is presented in Singpurwalla and Soyer [14]. Perhaps the most important aspect of these models is related to the stochastic structure of the underlying failure process. This could be a "times-between-failures" model which assumes that the times between successive failures follow a specific distribution whose parameters depend on the number of faults remaining in the program after the most recent failure. One of the most celebrated failure models in this group is that of Jelinski and Moranda [4] where the basic assumption is that there are a fixed number of initial faults in the software and each fault causes failures according to a Poisson process with the same failure rate λ. After each failure, the fault causing the failure is detected and removed with certainty so that the total number of faults in the software is decreased by one.

The available models on software failure in the literature do not apply to the specific structure of our operational process or the operational profile. They do not provide any distinction with respect to the operations performed by the system. This, however, is an important condition that can not be overlooked for realistic applications. In this section, we introduce a software failure model that incorporate the operational profile.

Most of the well-known models for assessing software reliability are centered around the interfailure times or the point processes generated by software failures. Let T_n be the time between the $(n-1)$th and the nth software failure and $U_n = T_1 + T_2 + \cdots + T_n$ be the time of the nth failure. The model of Jelinski and Moranda [4] assumes that the failure rate of T_n is a constant, proportional to the number of faults remaining in the program. More precisely, T_n has the exponential distribution

$$P[T_n > t] = \mathrm{e}^{-k\lambda t} \tag{12}$$

if there are k faults remaining in the software.

We now construct the following extension of the Jelinski–Moranda model such that

$$P[T_n > t | Y_s = i;\ U_{n-1} \leqslant s \leqslant U_{n-1} + t] = \mathrm{e}^{-k\lambda(i)t}, \tag{13}$$

where k is the number of faults remaining in the software, as before, and $\lambda(i)$ is the rate of software failures generated by each fault during operation $i \in E$. Note that this model considers not only the sequence of operations the software performs, but the amount of time for each operation as well.

The Jelinski–Moranda model makes several assumptions which can be criticized. It assumes that faults are equivalent in the sense that they all contribute the same amount to the failure rate. In reality different faults will differ in importance and so have a different effect on the failure rate. It also assumes that at each failure there is perfect debugging and no new errors are created; thus, the successive software failure rates are decreasing. During the debugging process new bugs may be created as a result of the changes to the program. Furthermore, the input space, and therefore, the operational profile may change as a result of the changes made to the program.

Another important issue in modelling software failures concern the relationship between faults and failures of operations as specified by the design of the software system. In some cases, each fault may be associated with only one operation so that only that fault can cause the failure of that operation. This model is referred to as the *individual faults* model by Özekici et al. [12]. If, for example, the design of the software involves a modular structure without interaction between the modules, then it may be more

appropriate to use such models. This is true in particular if each operation executes a specific module and causes failures due to possible faults within that module. Another model assumes that any fault can cause the failure of any operation of the software. Therefore, each operation can access any fault in the whole software and this is referred to as the *common faults* model. The model that should be used depends on the design of the software system. 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. It is implicit in (13) that our model assumes common faults since each fault generates a failure process with rate $\lambda(i)$ during operation i. This simplifies our construction because we only need to keep track of the number of the common faults remaining as time goes on. Otherwise, the construction will require another process for each type of fault in the system. We should point out that this presents only dimensional difficulties in computations.

In dealing with software reliability, one is interested in the number of faults N_t remaining in the software at time t. Then, N_0 is the initial number of faults and the process $N = \{N_t;\ t \geqslant 0\}$ depicts the stochastic evolution of the number of faults. If there is perfect debugging, then N decreases as time goes on, eventually to diminish to zero. Defining the bivariate process $Z_t = (Y_t, N_t)$, the basic failure model now becomes

$$P[T_n > t | Z_s = (Y_s, N_s) = (i,k);\ U_{n-1} \leqslant s \leqslant U_{n-1} + t] = \mathrm{e}^{-k\lambda(i)t}. \tag{14}$$

An alternate way of modelling the software failure process is to use the counting process M to count failures where M_t is the number of failures of the software that are observed until time t. In our case, with perfect debugging, it is clear that $M_t = N_0 - N_t$. Note that the point process M is modulated by $Z = (Y, N)$ such that the intensity function $\hat{\lambda}(t)$ at time t is

$$\hat{\lambda}(t) = N_t \lambda(Y_t). \tag{15}$$

This presents an interesting resemblance to the intensity function of Markov modulated Poisson processes with intensity function

$$\hat{\lambda}(t) = \lambda(Y_t), \tag{16}$$

where the modulating process is Y. A survey on such processes can be found in Fischer and Meier-Hellstern [3]. These processes are used in connection to queueing models where in each state i of the Markov process Y, customers arrive according to a Poisson process with arrival rate $\lambda(i)$. Another general type of model that is noteworthy in this context is the doubly stochastic Poisson process introduced by Kingman [5] and studied by many others. The Markov modulated Poisson model is a special doubly stochastic Poisson model where the intensity process is as given by (16). It is well-known that, in these models, the counting process M is conditionally a nonstationary Poisson process given the modulating process Y. One can thus use the Poisson distribution and its properties in the analysis.

In our more general setting, as depicted by (15), this is not necessarily true and M is not a Markov modulated Poisson process. It is clear that the Markov modulated Poisson model (16) is a special case of our model (15) where there is no debugging so that $N_t = N_0$ for all $t \geqslant 0$.

4. Markovian analysis

In this section, we suppose that the main parameters of the model (that is, the vectors λ, μ and the matrix P) are all known. Our construction of the process $Z = (Y, N)$ implies that it is a Markov process with discrete state space $F = E \times \{0, 1, 2, \ldots\}$. This follows by noting that Y is a Markov process and N is a

process that decreases by 1 after an exponential amount of time with a rate that depends only on the state of Y. In particular, if the current state of Z is (i,n) for any $n>0$, then the next state is either (j,n) with rate $\mu(i)P(i,j)$ or $(i,n-1)$ with rate $n\lambda(i)$. If $n=0$, then the next state is $(j,0)$ with rate $\mu(j)$. Note that 0 is an absorbing state for N.

This implies that the sojourn in state (i,n) is exponentially distributed with rate

$$\beta(i,n)=\mu(i)+n\lambda(i) \tag{17}$$

and the generator Q of Z is

$$Q((i,n),(j,m))=\begin{cases}-(\mu(i)+n\lambda(i)), & j=i,\ m=n,\\ \mu(i)P(i,j), & j\neq i,\ m=n,\\ n\lambda(i), & j=i,\ m=n-1.\end{cases} \tag{18}$$

Defining the transition function of Z as

$$\widetilde{P}_{(i,n)(j,m)}(t)=P[Y_t=j,\ N_t=m|Y_0=i,\ N_0=n] \tag{19}$$

we observe that it has the matrix exponential form

$$\widetilde{P}(t)=\mathrm{e}^{tQ}=\sum_{k=0}^{+\infty}\frac{t^k}{k!}Q^k. \tag{20}$$

The distribution of N_t can now be obtained through

$$P[N_t=m|Y_0=i,\ N_0=n]=\sum_{j\in E}\widetilde{P}_{(i,n)(j,m)}(t). \tag{21}$$

In particular, this gives the probability that the software is perfect at time t when we take $m=0$. Note that unconditional probabilities and expectations can be computed using the distribution of (Y_0,N_0). For example,

$$P[N_t=m]=\sum_{i\in E}\sum_{n=0}^{+\infty}P[Y_0=i,\ N_0=n]P[N_t=m|Y_0=i,\ N_0=n] \tag{22}$$

for any m and t. This observation holds true for all quantities of interest that we discuss in the following analysis.

Reliability is defined as the probability of failure free operation for a specified time. We will denote this by the function

$$R(i,n,t)=P[T_1>t|Y_0=i,\ N_0=n]=P[N_t=n|Y_0=i,\ N_0=n] \tag{23}$$

defined for all $(i,n)\in F$ and $t\geqslant 0$. Note that this is equal to the probability that there will be no arrivals until time t in a Markov modulated Poisson process with intensity function $\hat{\lambda}(t)=n\lambda(Y_t)$. Thus, using the matrix generating function (22) in Fischer and Meier-Hellstern [3] with $z=0$, we obtain the explicit formula

$$R(i,n,t)=\sum_{j\in E}\mathrm{e}^{(A-n\Lambda)t}(i,j), \tag{24}$$

where

$$\Lambda(i,j)=\begin{cases}\lambda(i), & \text{if } j=i,\\ 0, & \text{if } j\neq i,\end{cases} \tag{25}$$

and $\mathrm{e}^{(A-n\Lambda)t}$ is the exponential matrix

$$\mathrm{e}^{(A-n\Lambda)t}=\sum_{k=0}^{+\infty}\frac{t^k}{k!}(A-n\Lambda)^k. \tag{26}$$

Integrating (24), we obtain the following system of linear equations for the mean time to failure $f(i,n) = E[T_1|Y_0 = i,\ N_0 = n]$

$$f(i,n) = \frac{1}{n\lambda(i)} + \sum_{j\in E} A(i,j) f(j,n) \tag{27}$$

which can be solved easily for any given $n > 0$. It is clear that $f(i,0) = +\infty$. As a matter of fact, if we obtain the solution $g(i) = f(i,1)$ for $n = 1$ by solving

$$g(i) = \frac{1}{\lambda(i)} + \sum_{j\in E} A(i,j) g(j) \tag{28}$$

then it follows from the same equation that

$$f(i,n) = \frac{g(i)}{n} \tag{29}$$

is the solution of (27). Note that this result is quite intuitive since if the failure rate is $n\lambda(i)$ during operation i, then it is only natural that the mean time to failure is $(1/n)$ of the mean time to failure when the failure rates are only $\lambda(i)$. The ratio of the failure rates is n irrespective of the operation.

The random time

$$L = \inf\{t \geqslant 0;\ N_t = 0\} \tag{30}$$

is an important quantity associated with the reliability of the software since it represents the time of perfection, i.e., the first time at which the software is completely free of faults. Now,

$$\begin{aligned} P[L > t|Y_0 = i,\ N_0 = n] &= P[N_t > 0|Y_0 = i,\ N_0 = n] = 1 - P[N_t = 0|Y_0 = i,\ N_0 = n] \\ &= 1 - \sum_{j\in E} \widetilde{P}_{(i,n)(j,0)}(t). \end{aligned} \tag{31}$$

If we let $h(i,n) = E[L|Y_0 = i,\ N_0 = n]$ denote the mean time to perfection, then the Markov property at the time of the first transition of (Y,N) gives

$$h(i,n) = \beta(i,n)^{-1}\left[1 + \mu(i)\sum_{j\in E} P(i,j)h(j,n) + n\lambda(i)h(i,n-1)\right] \tag{32}$$

for $n > 0$ which is a system of linear equations that can be solved recursively in n using the boundary condition $h(i,0) = 0$.

Our analysis in this section demonstrates how the Markovian structure of the model can be exploited to compute several quantities of interest on software reliability. This can be done with ease once the model parameters are identified. We now address some statistical issues regarding these parameters in case they are not known.

5. Bayesian analysis

The model and the Markovian analysis presented above are based on the assumption that the parameters are specified. In this section, we will present a Bayesian analysis of the model by describing the uncertainty about the parameters probabilistically via some prior distributions. In particular, we will assume independent gamma priors on each $\lambda(i)$ with shape parameter $a(i)$, and scale parameter $b(i)$, denoted as $\lambda(i) \sim \text{Gamma}\ (a(i), b(i))$ for all $i \in E$. Similarly, independent gamma priors are assumed for the components of μ as $\mu(i) \sim \text{Gamma}\ (c(i), d(i))$ for all $i \in E$. The prior for initial number of faults N_0 is assumed

to be a Poisson distribution with parameter γ, denoted as $N_0 \sim \text{Poisson}(\gamma)$. For the components of the transition matrix P, we assume that the ith row $P_i = \{P(i,j);\ j \in E\}$ has a Dirichlet prior

$$p(P_i) \propto \prod_{j \in E} P(i,j)^{\alpha_j^i - 1} \tag{33}$$

denoted as Dirichlet $\{\alpha_j^i;\ j \in E,\ j \neq i\}$ and P_i's are independent for all $i \in E$. Furthermore, it is assumed that apriori λ, μ, P and N_0 are independent. We define $\Theta = (N_0, P, \lambda, \mu)$ and denote the joint prior distribution of Θ by $p(\Theta)$.

5.1. Prior analysis

We note that the generator matrix A of Y and the generator matrix Q of Z are functions of Θ. Furthermore, the results presented in the Markovian analysis are conditional on the unknown vector Θ. Two quantities of interest in the analysis are the reliability function and the distribution of the time of perfection. We can write the reliability function (24) as

$$P[T_1 > t | Y_0 = i,\ \Theta] = \sum_{j \in E} \mathrm{e}^{(A(\Theta) - N_0 \Lambda(\Theta))t}(i,j). \tag{34}$$

Conditional on Θ, $\mathrm{e}^{(A(\Theta) - N_0\Lambda(\Theta))t}$ can be computed from the matrix exponential form using one of the available methods, for example, in Moler and van Loan [7]. Then prior to any operational testing, we can make reliability predictions as

$$P[T_1 > t | Y_0 = i] = \int P[T_1 > t | Y_0 = i,\ \Theta] p(\Theta)\, \mathrm{d}\Theta \tag{35}$$

which can be approximated via simulation as a Monte Carlo integral

$$P[T_1 > t | Y_0 = i] \approx \frac{1}{G} \sum_g P[T_1 > t | Y_0 = i,\ \Theta^{(g)}] \tag{36}$$

by generating G realizations from the prior distribution $p(\Theta)$.

Similarly, we can obtain prior predictive distribution of L. Conditional on Θ, distribution of L is given by

$$P[L > t | Y_0 = i,\ \Theta] = 1 - \sum_{j \in E} \widetilde{P}_{(i,N_0)(j,0)}(t, \Theta), \tag{37}$$

where $\widetilde{P}(t,\Theta)$ is the transition function (20) of the Z process which is a function of Θ. Conditional on Θ, $\widetilde{P}(t,\Theta)$ can be computed from the matrix exponential form using available methods. Then the prior predictive distribution of L is given by

$$P[L > t | Y_0 = i] = \int P[L > t | Y_0 = i,\ \Theta] p(\Theta)\, \mathrm{d}\Theta \tag{38}$$

which can be approximated as

$$P[L > t | Y_0 = i] \approx \frac{1}{G} \sum_g P[L > t | Y_0 = i,\ \Theta^{(g)}] \tag{39}$$

by generating G realizations from the prior distribution $p(\Theta)$.

5.2. Posterior analysis

Assume that during the operational phase the software is used for τ units of time during which K operations are performed and, of those, $K-1$ are completed. In the operational phase, debugging is performed and the failure times during each operation as well as the operation types and their durations are observed. The observed data from Koperations is given by

$$\mathscr{D} = \{(X_k, S_k),\ (U_1^k, U_2^k, \ldots, U_{M_k}^k);\ k = 1, \ldots, K\}, \tag{40}$$

where U_j^k is the time (since the start of the kth operation) of jth failure during the kth operation. Moreover, X_k is the kth operation performed while S_k is the time at which it starts.

To simplify the notation somewhat, we will let $N_k = N_{S_k}$ denote the total number of faults remaining in the software just before the kth operation. Similarly, M_k denotes the number of failures observed during the kth operation. Therefore,

$$N_{k+1} = N_k - M_k = N_1 - \sum_{j=1}^{k} M_j \tag{41}$$

since there is perfect debugging of any failure. Note that our notation also implies $N_1 = N_0$. Moreover, we suppose that the initial operation is $X_1 = i$ for some operation i and it starts at $S_1 = 0$. Since the results presented in previous sections are all conditional on some initial state i, the posterior analysis will be based on this arbitrary initialization.

The likelihood function of $\Theta = (N_0, P, \lambda, \mu)$ given data is

$$\mathscr{L}(\Theta|\mathscr{D}) = \mathscr{L}_K \left\{ \prod_{k=1}^{K-1} P(X_k, X_{k+1}) \mu(X_k) \mathrm{e}^{-\mu(X_k)(S_{k+1}-S_k)} \frac{N_k!}{(N_k - M_k)!} \lambda(X_k)^{M_k} \mathrm{e}^{-\lambda(X_k)\left[\sum_{j=1}^{M_k} U_j^k + (N_k - M_k)(S_{k+1}-S_k)\right]} \right\}, \tag{42}$$

where $\mathscr{L}_K$ is the contribution of the Kth operation to the likelihood given by

$$\mathscr{L}_K = \mathrm{e}^{-\mu(X_K)(\tau - S_K)} \frac{N_K!}{(N_K - M_K)!} \lambda(X_K)^{M_K} \mathrm{e}^{-\lambda(X_K)\left[\sum_{j=1}^{M_K} U_j^K + (N_K - M_K)(\tau - S_K)\right]}. \tag{43}$$

Given the independent priors, the posterior distribution of P_i's can be obtained as independent Dirichlets given by

$$(P_i|\mathscr{D}) \sim \text{Dirichlet}\left\{ \alpha_j^i + \sum_{k=1}^{K-1} 1(X_k = i,\ X_{k+1} = j);\quad j \in E \right\}, \tag{44}$$

where $1(.)$ is the indicator function. Similarly, the posterior distributions of $\mu(i)$'s are obtained as independent gamma densities given by

$$(\mu(i)|\mathscr{D}) \sim \text{Gamma}\left(c(i) + \sum_{k=1}^{K-1} 1(X_k = i),\ d(i) + \sum_{k=1}^{K} (S_{k+1} - S_k) 1(X_k = i) \right), \tag{45}$$

where $S_{K+1} = \tau$. We note that posteriori μ and P are independent of λ and N_0 as well as each other.

A tractable Bayesian analysis for λ and N_0 is not possible due to the infinite sums involved in the posterior terms, but a fully Bayesian analysis can be made by using a Gibbs sampler. Kuo and Yang [6] provide an application of Gibbs sampling in software reliability. The implementation of the Gibbs sampler requires the full posterior conditionals $p(N_0|\lambda, \mathscr{D})$ and $p(\lambda(i)|\lambda^{(-i)}, N_0, \mathscr{D})$ for all $i \in E$ where $\lambda^{(-i)} = \{\lambda(j);\ j \neq i,\ j \in E\}$. We again note the independence of the full conditionals on μ and P.

We can write

$$p(N_0|\lambda,\mathscr{D}) \propto \prod_{k=1}^{K} \frac{N_k!}{(N_k-M_k)!} \exp\left(-\lambda(X_k)\left[\sum_{j=1}^{M_k} U_j^k + (N_k-M_k)(S_{k+1}-S_k)\right]\right) \frac{\mathrm{e}^{-\gamma}\gamma^{N_1}}{N_1!}. \tag{46}$$

Using the fact that $N_1 = N_0$ and $(N_n - M_n) = N_1 - M$, we recognize

$$\prod_{k=1}^{K} \frac{N_k!}{(N_k-M_k)!}\frac{1}{N_1!} = \frac{1}{(N_n-M_n)!} = \frac{1}{(N_1-M)!}, \tag{47}$$

where $M = \sum_{k=1}^{K} M_k$ and obtain

$$p(N_1|\lambda,\mathscr{D}) \propto \frac{\left(\gamma \mathrm{e}^{-\sum_{k=1}^{K}\lambda(X_k)(S_{k+1}-S_k)}\right)^{N_1-M}}{(N_1-M)!} \tag{48}$$

implying

$$(N_0 - M|\lambda,\mathscr{D}) \sim \text{Poisson}\left(\gamma \mathrm{e}^{-\sum_{k=1}^{K}\lambda(X_k)(S_{k+1}-S_k)}\right). \tag{49}$$

For the full conditionals, $p(\lambda(i)|\lambda^{(-i)}, N_0, \mathscr{D})$'s are obtained as

$$(\lambda(i)|\lambda^{(-i)}, N_0, \mathscr{D}) \sim \text{Gamma}\left(a(i) + \sum_{k=1}^{K} M_k 1(X_k = i),\ b(i) + \sum_{k=1}^{K} W_k 1(X_k = i)\right), \tag{50}$$

where $W_k = \sum_{j=1}^{M_k} U_j^k + (N_k - M_k)(S_{k+1} - S_k)$.

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 N_0, aposteriori the $\lambda(i)$'s are independent.

Once our uncertainty about Θ is revised to $p(\Theta|\mathscr{D})$, then we are interested in making posterior reliability predictions as $P[T > t|\mathscr{D}]$. Note that conditional on Θ, using the Markov property of the Z process, we can write

$$P[Y_{\tau+t} = j,\ N_{\tau+t} = m|\Theta, Y_u, N_u;\ u \leqslant \tau] = P[Y_{\tau+t} = j,\ N_{\tau+t} = m|\Theta, Y_\tau, N_\tau]. \tag{51}$$

Thus, given the data $\mathscr{D}$

$$P[Y_{\tau+t} = j,\ N_{\tau+t} = m|\Theta,\mathscr{D}] = P[Y_{\tau+t} = j,\ N_{\tau+t} = m|\Theta, N_\tau, Y_\tau] \tag{52}$$

and the transition function is given by

$$\widetilde{P}_{(X_K, N_0-M)(j,m)}(t,\Theta) = P[Y_{\tau+t} = j,\ N_{\tau+t} = m|\Theta, N_\tau = N_0 - M, Y_\tau = X_K], \tag{53}$$

where $\widetilde{P}(t,\Theta)$ is the transition function (20) of the Z process as a function of Θ. Also, conditional on Θ we can write

$$P[T > t|\Theta,\mathscr{D}] = P[N_{\tau+t} = N_0 - M|\Theta, N_\tau = N_0 - M, Y_\tau = X_K] \tag{54}$$

and using the result of Fischer and Meier-Hellstern [3] we obtain

$$P[T > t|\Theta,\mathscr{D}] = \sum_{j\in E} \mathrm{e}^{(A(\Theta)-(N_0-M)\Lambda(\Theta))t}(X_K, j). \tag{55}$$

Similarly, letting $\hat{L} = L - \tau$ denote the remaining time to perfection

$$P[\hat{L} > t|\Theta,\mathscr{D}] = P[\hat{L} > t|\Theta, N_\tau, Y_\tau] \tag{56}$$

we obtain

$$P[\hat{L} > t|\Theta, \mathscr{D}] = P[\hat{L} > t|\Theta, N_\tau = N_0 - M, Y_\tau = X_K] = 1 - \sum_{j\in E} \widetilde{P}_{(X_K, N_0-M)(j,0)}(t, \Theta) \tag{57}$$

from (37).

Now given (55) and (57), we can obtain posterior analogs of the prior predictions (35) and (38). We can obtain the posterior reliability prediction as

$$P[T > t|\mathscr{D}] = \int P[T > t|\Theta, \mathscr{D}]p(\Theta|\mathscr{D})\,\mathrm{d}\Theta \tag{58}$$

which can be approximated using (55) as a Monte Carlo integral

$$P[T > t|\mathscr{D}] \approx \frac{1}{G}\sum_g P[T > t|\Theta^{(g)}, \mathscr{D}] \tag{59}$$

using G realizations from the posterior distribution $p(\Theta|\mathscr{D})$. Similarly, the posterior predictive distribution of $\hat{L}$ is given by

$$P[\hat{L} > t|\mathscr{D}] = \int P[\hat{L} > t|\Theta, \mathscr{D}]p(\Theta|\mathscr{D})\,\mathrm{d}\Theta \tag{60}$$

which can be approximated using (57) as

$$P[\hat{L} > t|\mathscr{D}] \approx \frac{1}{G}\sum_g P[\hat{L} > t|\Theta^{(g)}, \mathscr{D}] \tag{61}$$

using G realizations from the posterior distribution $p(\Theta|\mathscr{D})$.

6. Numerical illustration

To illustrate our results, we consider a situation similar to the example given in Table 1 and assume a scenario with three operations. These can be thought as the more frequent operations given in Table 1. Thus, in our notation we have state space of $E = \{1, 2, 3\}$ for the operational process Y with the generator matrix given by

$$A = \begin{bmatrix} -\mu(1) & \mu(1)P(1,2) & \mu(1)P(1,3) \\ \mu(2)P(2,1) & -\mu(2) & \mu(2)P(2,3) \\ \mu(3)P(3,1) & \mu(3)P(3,2) & -\mu(3) \end{bmatrix}. \tag{62}$$

Since there is no actual data available, the probabilistic structure presented in Section 3 for the software failure process was simulated. In so doing, it was assumed that the software was used about $\tau = 970$ units of time during which $(K-1) = 99$ operations were completed assuming that the initial state was $X_1 = 1$ and data as in (40) was obtained. The simulated data is not given here due to space limitations.

In our illustration we assume diffused prior distributions for the elements of Θ. More specifically, for the Dirichlet priors in (33) we assume that $\alpha_j^i = 1$ for all $(i, j;\ j \neq i)$, for the gamma priors of $\lambda(i)$'s we assume $a(i) = b(i) = 0.1$ and for $\mu(i)$'s $c(i) = d(i) = 0.1$ for all $i \in E$. Furthermore, we assume that apriori the expected number of faults in the software is 150, that is, $\gamma = 150$ in the Poisson prior of N_0. This choice of parameters represents a high degree of prior uncertainty about components of Θ. In a situation where prior information exists, the prior parameters can be specified by eliciting best guess values for elements of Θ and uncertainties about these values from software engineers. In what follows, we will focus on posterior analysis of the simulated data.

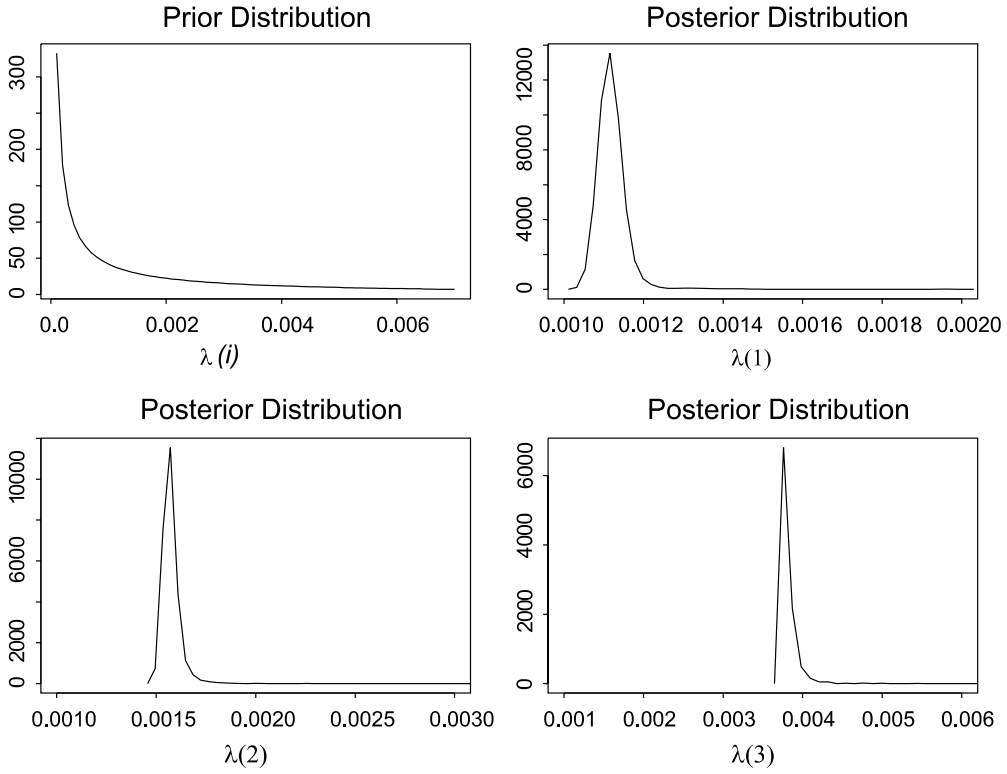

Fig. 1. Prior and posterior distributions of $\lambda(i)$'s.

The posterior distributions of P_i's can be obtained as Dirichlet distributions given by (44) and the posterior distributions of $\mu(i)$'s are obtained as gamma distributions as in (45). As discussed in Section 5.2, the posterior distributions of $\lambda(i)$'s and N_0 are obtained by implementation of the Gibbs sampler. In Fig. 1 we present the common prior distribution and the posterior distributions for $\lambda(i)$'s. The density plots that are presented for the posterior distributions are based on the samples from the joint distribution of $\lambda(i)$'s obtained via the Gibbs sampler. The figure shows the effect of the data on each of the $\lambda(i)$'s. We note that compared to the prior distribution, the posterior distributions are very peaked and they are concentrated in different regions. In fact, the posterior means of the $\lambda(i)$'s are very close to the actual values used in simulating the data.

The prior and posterior distributions of N_0 are shown in Fig. 2. The posterior distribution is lot more peaked than the prior and is concentrated around higher values. The posterior mean of N_0 is approximately 193 whereas the prior mean was 150.

Once the posterior samples are available we can make posterior reliability predictions using the Monte Carlo approximation (59). In the simulated data the last operation observed was of type 2, that is, in (55), we have $X_K = 2$. Evaluation of (59) requires approximating the matrix exponential form $e^{(A(\Theta)-(N_0-M)\Lambda(\Theta))t}$ for each realization of Θ from the posterior distribution. In our evaluation, we use the approximation

$$e^{Ht} \approx \left(I - H\frac{t}{z}\right)^z \tag{63}$$

for z large where $H = (A(\Theta) - (N_0 - M)\Lambda(\Theta))$. In Fig. 3 we show the posterior predictive reliability function and compare it with the actual reliability function based on the values of Θ used in simulating the

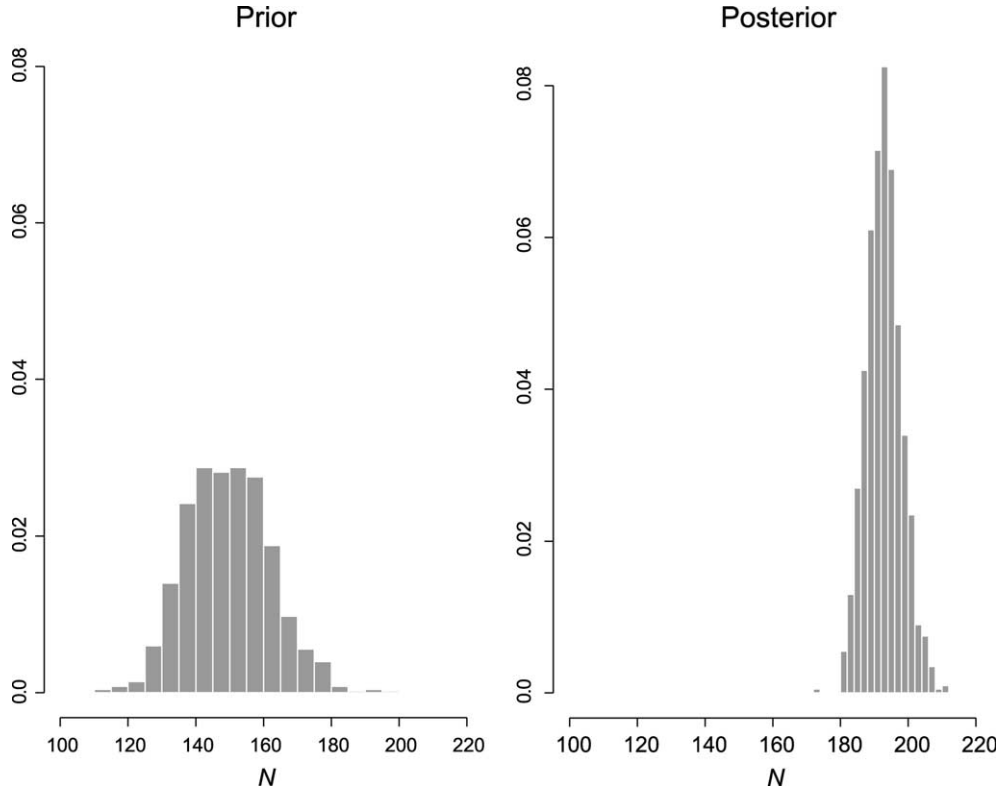

Fig. 2. Prior and posterior distributions of N_0.

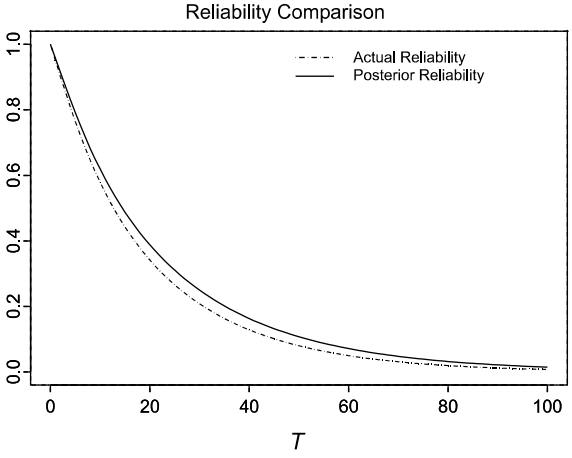

Fig. 3. Comparison of actual and posterior reliability functions.

failure process. We note that the posterior reliability function is very close to the actual reliability function. In the Bayesian setup we can also make probability statements about reliability at a given mission time t; that is, given data we can make probability statements about $R(X_K, N_0 - M, t)$ for fixed t.

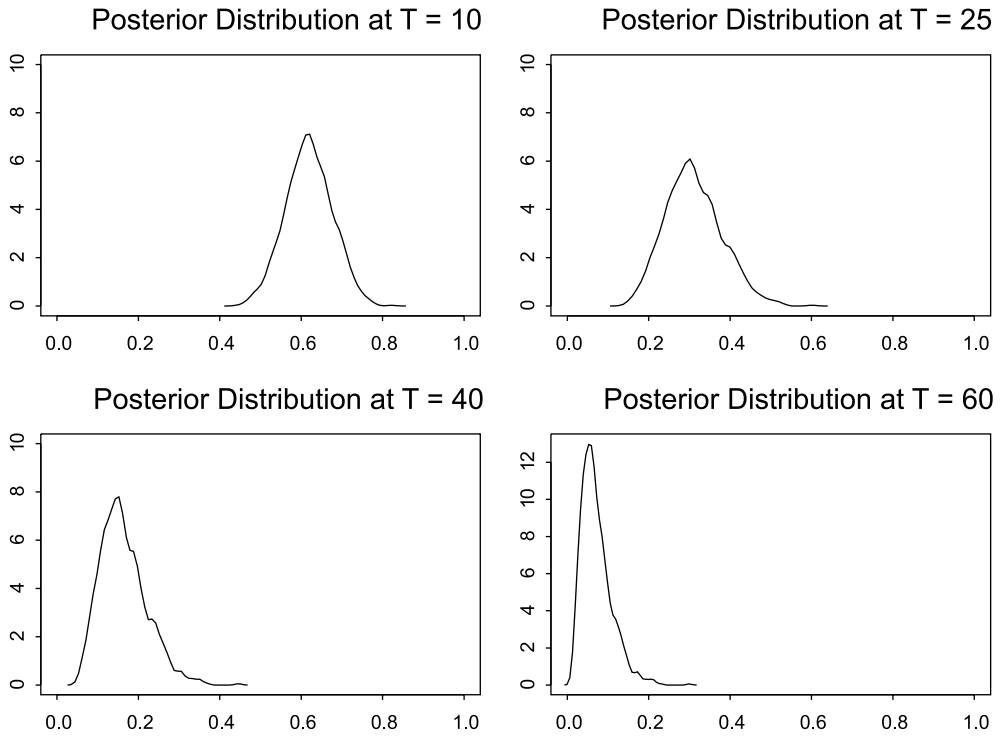

Fig. 4. Posterior distributions of reliability for different mission times T.

This simply requires that we look at the posterior distribution of the random variable $e^{(\Lambda(\Theta)-(N_0-M)\Lambda(\Theta))t}$ using the posterior samples of Θ. In Fig. 4 we show the posterior distribution of reliability at mission times $t = 10$, 25, 40 and 60. At mission time 10, the posterior distribution of reliability is concentrated around 0.6. As expected the posterior distribution of the reliability shifts to the left as mission time increases. We note that the posterior reliability function presented in Fig. 3 represents the means of the posterior reliability distributions at different mission times.

Acknowledgements

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

References

[1] E. Çınlar, Introduction to stochastic processes, Prentice-Hall, Englewood Cliffs, NJ, 1975.
[2] E. Çınlar, S. Özekici, Reliability of complex devices in random environments, Probability in the Engineering and Informational Sciences 1 (1987) 97–115.
[3] W. Fischer, K. Meier-Hellstern, The Markov-modulated Poisson process cookbook, Performance Evaluation 18 (1992) 149–171.
[4] Z. Jelinski, P. Moranda, Software reliability research, in: W. Freiberg (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972, pp. 465–484.
[5] J.F.C. Kingman, On doubly stochastic Poisson processes, Proceedings of the Cambridge Philosophical Society 60 (1964) 923–960.

[6] L. Kuo, T.Y. Yang, Bayesian computations of software reliability, Journal of Computational and Graphical Statistics 4 (1995) 65–82.
[7] C. Moler, C. van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Review 20 (1978) 801–836.
[8] J.D. Musa, Operational profiles in software reliability engineering, IEEE Software 10 (1993) 14–32.
[9] J.D. Musa, The operational profile, in: S. Özekici (Ed.), Reliability and Maintenance of Complex Systems, NATO ASI Series, vol. F154, Springer, Berlin, 1996, pp. 332–343.
[10] J.D. Musa, A. Iannino, K. Okumoto, Software reliability: Measurement, prediction, application, McGraw-Hill, New York, 1987.
[11] S. Özekici, Complex systems in random environments, in: S. Özekici (Ed.), Reliability and Maintenance of Complex Systems, NATO ASI Series, vol. F154, Springer, Berlin, 1996, pp. 133–157.
[12] S. Özekici, İ.K. Altınel, S. Özçelikyürek, Testing of software with an operational profile, Naval Research Logistics 47 (2000) 620–634.
[13] N.D. Singpurwalla, T.A. Mazzuchi, S. Özekici, R. Soyer, Software reliability and the operational profile, Tech. report, GWU/IRRA Series TR-99/1, The George Washington University, Washington, DC 20052, 1999.
[14] N.D. Singpurwalla, R. Soyer, Assessing the reliability of software: An overview, in: S. Özekici (Ed.), Reliability and Maintenance of Complex Systems, NATO ASI Series, vol. F154, Springer, Berlin, 1996, pp. 345–367.
[15] J.A. Whittaker, J.H. Poore, Markov analysis of software specifications, ACM Transactions on Software Engineering and Methodology 2 (1993) 93–106.
[16] J.A. Whittaker, M.G. Thomason, A Markov chain model for statistical software testing, IEEE Transactions on Software Engineering 20 (1994) 812–824.
[17] C. Wohlin, P. Runeson, Certification of software components, IEEE Transactions on Software Engineering 20 (1994) 494–499.