Rev.R.Acad.Cien.Exact.Fis.Nat. (Esp)
Vol. 92, N.º 4, pp 323-328, 1998
Monográfico: Problemas complejos de decisión

BAYESIAN INFERENCE AND OPTIMAL RELEASE TIMES FOR TWO SOFTWARE FAILURE MODELS

(software reliability/Jelinski-Moranda model/Littlewood's nonhomogeneous poisson process/Markov chain Monte Carlo/ expected utility/optimal release)

M. P. Wiper*, D. Ríos Insua** and R. Hierons***

* Departamento de Estadística y Econometría, Universidad de Carlos III de Madrid, C/ Madrid 126, 28903 Getafe (Madrid).

** Departamento de Ciencias Experimentales e Ingeniería, Universidad Rey Juan Carlos, Tulipán s/n, 28933 Móstoles (Madrid).

*** Department of Mathematical and Computing Sciencies. Goldsmitns College. University of London.

ABSTRACT

We carry out Bayesian inference for the Jelinski-Moranda and Littlewood software failure models given a sample of failure times. Furthermore, we illustrate how to assess the optimal length of an additional pre-release testing period under each of these models. Modern Bayesian computational methods are used to estimate the posterior expected utility of testing for and additional time.

1. INTRODUCTION

Software usually goes through various stages of testing before its eventual release. Early stage testing might be carried out in order to try modifications of software code being developed, or to assess the progress of the development procedure. At this stage, random or partition testing procedures are often, used, whereby inputs are randomly generated from the operational profile (assumed distribution of usage) of the code and the number of faulty outputs are recorded. See e.g. Chen & Yu (3) or Hierons & Wiper (8), for introductions. Pre-release testing is undertaken both to detect and correct bugs in a piece of code so that it is unlikely to fail in service. Such failures could lead to expensive recalls of the product and loss in customer confidence. In this case, interest centres on the CPU time between failures and intra-failure times are recorded in testing. This is the scenario we consider here.

Pre-release testing can be expensive, both in terms of the costs involved in the procedure (assessment of failure, assessment of a realistic operational profile,...) and of the opportunity costs of non-release of the software. Short test periods leading to early release are particularly important in a competitive market, when different companies produce similar software products, and the first to release may well corner the market despite its product being less reliable than those of its competitors. On the other hand, the release of «buggy», untested code could have serious financial implications, when customers cease to buy the company's products. In particular, safety critical software needs to be very reliable and careful testing is likely to be of much greater importance than early release, see e.g. Littlewood & Wright (15). It is important to develop procedures to decide upon suitable time periods for software testing.

Different criteria have been applied to the problem of when to stop software testing. For example, optimization of cost criteria subject to given reliability constraints, e.g. (21), (10), or game-theoretic techniques (22). Decision theoretic approaches are considered by Dalal & Mallows (4) and Singpurwalla (18). Singpurwalla looks at preposterior analysis, i.e. deciding the length of a test period before doing any testing. In this paper, we shall consider the case where we have already done some testing and then wish to decide the length of a possible final test period prior to software release.

Unless we wish to consider non-parametric approaches, in order to assess what length of test to use, we need to have a model for how failures occur in software. There are many different models for software failure in the literature, see e.g. Dale (5) or Singpurwalla & Wilson (19) for thorough reviews. Here we shall consider two of the most popular, those of Jelinski & Moranda (9) and Littlewood (12). We will introduce these models in Section 2.

In Sections 2.1 and 2.2, we assume that testing has taken place (either for a fixed time, or until a certain number of failures were observed) and use Bayesian inference to estimate the posterior distributions of the relevant parameters of both models. In particular, we show how to estimate the number of bugs remaining in the software after the test period.

In Section 3, we suggest a utility function and show how the posterior expected utility of a further test period of length T may be estimated. An example using software failure data is given in Section 4. In Section 5, we briefly illustrate how a recently introduced approach of Bielza et al (2) may be applied to the related problem of preposterior testing. In Section 6 we draw some conclusions and suggest extensions of our approach.

2. INFERENCE FOR THE JELINSKI-MORANDA AND LITTLEWOOD MODELS

We introduce the two software models which we shall consider in this paper. Let T_1, T_2,... be the successive times between failures of a piece of software under test. The Jelinski-Moranda (JM) model (9) assumes that:

$$T_i|N,\ \phi \sim \mathcal{E}\left((N-i+1)\,\phi\right)$$

i.e. that intra-failure times are exponentially distributed. The model basically assumes that, initially, there are N faults in the software and after each one is discovered, it is perfectly corrected. The parameter ϕ relates to the «size» of a fault.

Various criticisms of this model, see e.g. (13), have lead to several alternative models. One of them is Littlewood's (12), which assumes that

$$f\left(t_i|N,A,B,t_1,\ldots,t_{i-1}\right)=(N-i+1)A\frac{\left(B+t'_{i-1}\right)^{(N-i+1)A}}{\left(B+t'_i\right)^{(N-i+1)A+1}}$$

where $t'_i = \sum_{j=1}^{i} t_j$. Here, as with JM, N represents the number of bugs initially present in the code. For a fuller discussion of the significance of the other parameters and the relationship of the model to JM, see (13).

In the following subsections, we shall undertake Bayesian inference for both models. In both cases, we shall assume that either testing has been carried out until a fixed number m of failures have been observed, or that testing has been carried out for a fixed time t and that $m - 1$ failures have been observed before this time. In this second case, the m-th failure time t_m is thus considered to be right-censored. We look first at inference for the JM model.

2.1. JM

Bayesian inference for this model has also been examined in e.g. (14), (4) and (11). Here we just give a brief introduction. Assume that we test until we observe m failures. Then, the likelihood function is

$$L(N,\ \phi|\text{data}) \propto \frac{N!}{(N-m)!}\ \phi^m \exp\left(-\left([N+1]\,m\bar{t}-\sum_{i=1}^{m} it_i\right)\phi\right)$$

where $\bar{t} = \frac{1}{m}\sum_{i=1}^{m} t_i$. In the case where the last observation is right censored, the likelihood is as above, with m replaced by $m - 1$ in the first two terms.

We shall consider the following prior distributions for N and ϕ:

$$N \sim \mathcal{TP}(\lambda)$$

a Poisson distribution with mean λ, truncated at some upper limit, say N^*, and

$$\phi \sim \mathcal{G}(\alpha,\ \beta)$$

a gamma distribution. Under this model, given the full data, the marginal posterior distributions of both parameters may be written down:

$$P(N\ =\ n|\text{data}) \propto \frac{\lambda^{n-m}}{(n-m)!}\left(n+1+\frac{\beta-\sum_{i=1}^{m} it_i}{m\bar{t}}\right)^{-(\alpha+m)}$$

for $N \geq m$,

$$f(\phi|\text{data}) \propto \phi^{m+\alpha-1}\exp\left(-\lambda e^{-m\bar{t}\phi}-\left(\beta+m\bar{t}+m^2\bar{t}-\sum_{i=1}^{m} it_i\right)\phi\right)$$

Similar expressions for these posteriors are available, assuming that the last failure time is right truncated.

It is relatively straightforward to calculate moments of these distributions by either numerical integration, or truncation of sums methods. For example,

$$E[N|\text{data}]=\sum_{n=0}^{N^*} n\frac{\frac{\lambda^{n-m}}{(n-m)!}\left(n+1+\frac{\beta-\sum_{i=1}^{m} it_i}{m\bar{t}}\right)^{-(\alpha+m)}}{\sum_{j=0}^{N^*}\frac{\lambda^{j-m}}{(j-m)!}\left(j+1+\frac{\beta-\sum_{i=1}^{m} it_i}{m\bar{t}}\right)^{-(\alpha+m)}}$$

2.2. Littlewood

Inference for this model has also been undertaken by e.g. Littlewood (12) and Abdel Ghaly (1). We undertake a fully Bayesian approach. The likelihood, under this model, is given by

$$L(N,\ A,\ B|\text{data}) \propto \frac{N!}{(N-m)!}\ A^m\prod_{i=1}^{m}\frac{\left(B+t'_{i-1}\right)^{(N-i+1)A}}{\left(B+t'_i\right)^{(N-i+1)A+1}}$$

in the case of complete data, where $t'_i = \sum_{j=1}^{i} t_j$. In case the final failure time is truncated,

$$L(N,\, A,\, B|\text{data}) \propto \frac{N!}{(N-m+1)!} A^{m-1}\left(B+t'_m\right)\prod_{i=1}^{m}\frac{\left(B+t'_{i-1}\right)^{(N-i+1)A}}{\left(B+t'_i\right)^{(N-i+1)A+1}}.$$

We consider the following prior distributions:

$$N \sim \mathcal{TP}(\lambda)$$

as for JM

$$A \sim \mathcal{G}(\alpha,\, \beta),$$

i.e., a gamma distribution.

$$B\big|c \sim F_d^d$$

i.e. B has a distribution proportional to a scaled (Fisher's) F distribution with d degrees of freedom, both in the numerator and the denominator.

Given these priors, it is impossible to write down the full, marginal posterior distributions of the model parameters in simple form. Earlier authors, e.g. (12), (1) considered ad-hoc approaches to the inference problem. However, with the advent of modern Bayesian computational methods, we can simulate samples from the relevant posterior distributions. Specifically, we consider a (Metropoli within) Gibbs sampling approach, see e.g. (7), which proceeds as follows:

1. Set initial values $N^{(0)}$, $A^{(0)}$, $B^{(0)}$, $i = 0$
2. Generate $N^{(i+1)}$ from $f(N|A^{(i)}, B^{(i)}, \text{data})$
3. Generate $A^{(i+1)}$ from $f(A|N^{(i+1)}, B^{(i)}, \text{data})$
4. Generate $B^{(i+1)}$ from $f(B|N^{(i+1)}, A^{(i+1)}, \text{data})$
5. $i = i + 1$, goto 2.

The algorithm continues until convergence is judged to have been reached in practice, say after r iterations, when the sample $N^{(r+1)}, A^{(r+1)}, B^{(r+1)},\ldots, N^{(r+s)}, A^{(r+s)}, B^{(r+s)}$ approximates to a sample of size s from the joint posterior distribution.

The relevant posterior, conditional distributions needed for Gibbs sampling, given the full data, are as follows:

$$(N-m)\,|A,\, B,\, \text{data} \sim \mathcal{P}\left(\lambda\left(\frac{B}{B+t'_m}\right)^{a}\right) \qquad (1)$$

$$A\,|\,N,B,\text{data} \sim \mathcal{G}\left(\alpha+m,\beta-\sum_{i=1}^{m}(N-i+1)\log\frac{B+t'_{i-1}}{B+t'_i}\right) \qquad (2)$$

$$f(B|N,\, A,\, \text{data}) \propto \frac{B^{d/2-1}}{(c+B)^d}\prod_{i=1}^{m}\frac{\left(B+t'_{i-1}\right)^{(N-i+1)A}}{\left(B+t'_i\right)^{(N-i+1)A+1}} \qquad (3)$$

In the case where the last failure time is truncated, the posteriors are modified slightly: we replace $N - m$ by $N - m + 1$ in (1), replace $\alpha + m$ by $\alpha + m - 1$ in (2) and the posterior for B in (3) is multiplied by the extra term $\left(B+t'_m\right)$.

Sampling from the first two distributions is straightforward. To sample from the distribution of B, we use a Metropolis step, see e.g. (7): to generate $B^{(i+1)}$, we first generate a candidate B^{cand}, from a suitable candidate distribution. Specifically, we generate $B^{cand} \sim B^{(i)}\ F_e^e$. The degrees e of freedom of the F distribution can be adjusted to improve the sampling properties of the algorithm. Then accept the candidate and set $B^{(i+1)} = B^{cand}$ with probability (given the complete data)

$$p = \min\{1,\, P\} \text{ where}$$

$$P = \left(\frac{B^{cand}}{B^{(i)}}\right)^{d/2-1}\left(\frac{c+B^{(i)}}{c+B^{cand}}\right)^{d}\prod_{i=1}^{m}\left(\frac{B^{cand}+t'_{i-1}}{B^{(i)}+t'_{i-1}}\right)^{(N-i+1)A}\left(\frac{B^{(i)}+t'_i}{B^{cand}+t'_i}\right)^{(N-i+1)A+1}$$

where N and A are the current values of these parameters. Otherwise, reject the candidate and set $B^{(i+1)} = B^{(i)}$.

Features of the posterior distribution are then easy to estimate. For example, the posterior mean number of bugs initially in the software is estimated by

$$E[N|\text{data}] \approx \frac{1}{s}\sum_{i=r+1}^{r+s} N^{(i)}.$$

3. DEFINING A UTILITY FOR FURTHER TESTING

We assume now that after initial testing, we wish to decide upon an optimal time T for further testing. The costs of this further testing might depend upon the size of T (short test periods will be cheaper), the number of bugs discovered in testing (these will have to be corrected) and the number of bugs undiscovered (as the software will be released containing these bugs which could lead to future costs). Among the many possible utility functions, and, mainly for illustration purposes, we shall consider a utility function of the following form (assuming that we have already observed m failures):

$$U(T,\, \theta,\, t_{m+1},\ldots,\, t_{Nn}) = c_0 - c_1\frac{T}{T_{\max}} - c_2 Z(T) - c_3\left(N-m-Z(T)\right) \qquad (4)$$

where $Z(T)$ is the number of bugs discovered in the further test period, and θ are the relevant model parameters, $\theta(N, \phi)$ for JM and $\theta = (N, A, B)$ for Littlewood. T^{max} represents the maximum possible test period after which it would be considered worthless to release the software. Typically, we should expect that $c_2 < c_3$ as costs of correcting bugs pre-release are liable to be much less than costs of correction after release; we might even set $c_2 = 0$. The constant term c_0 is assumed positive and large enough to ensure that the utility function is always positive. The use of the constant term is unnecessary from the utility viewpoint, since it does not alter the optimal test time, but it is necessary for the approach we use later in Section 5. Other utility functions could include, for example, discount factors, opportunity costs, nonlinear terms in T, or the expected reliability of the code after testing See Singpurwalla (18) for further examples.

We wish to maximise in T the expected utility which is given by

$$\int U(T,\ \theta,\ t_{m+1},\dots,\ t_N)\, f(\theta|t_1,\dots,\ t_m)\, f(t_{m+1},\dots,\ t_N|\theta,\ t_1,\dots,\ t_m)\ dt_{m+1}\dots dt_N d\theta \tag{5}$$

For the JM model, the posterior expected utility can be evaluated exactly which is

$$c_0 - c_1\ \frac{T}{T^{max}}\ -\ (c_2 + c_3)\, E[Z(T)\ |\text{data}]\ -\ c_3 E[N\ |\text{data}]\ +\ c_3 m$$

with

$$E[Z(T)\ |\text{data}]\ =\ \frac{\sum \frac{(N-m)\lambda^{N-m}}{(N-m)!}\left[\upsilon^{-(\alpha+m)} - (\upsilon+T)^{-(\alpha+m)}\right]}{\sum_j \frac{\lambda^{j-m}}{(j-m)!}\left(\upsilon + (j-N)\, m\bar{t}\right)^{-(\alpha+m)}}$$

where

$$\upsilon\ =\ \beta + (N+1)\, m\bar{t}\ -\ \sum it_i$$

and $E(N|\text{data})$ is as in Section 2.1.

Under Littlewood's model we may approximate the expected utility from the Gibbs sampler output as

$$\frac{1}{s}\sum_{i=r}^{r+s} U\left(T,\ N^{(i)},\ A^{(i)},\ B^{(i)}\right)$$

In the following example we illustrate our methods for estimation of the number of faults remaining in software after testing and for choosing a final test period.

4. EXAMPLE

We consider software failure data taken from Table 2 in Littlewood [13]. Littlewood gives execution times in hundredths of seconds (*cs*) between failures. 86 failure times are recorded, with the total time to the last failure being 103334 *cs* and the time between the last two failures being 3902 *cs*.

For this problem, we assume that we observe all those data. The first question of interest is, given prior distributions, what is the posterior estimate of the number N of bugs initially in the software? Table I gives posterior estimates of N for both the JM and Littlewood models under optimistic and pessimistic priors for N. (Relatively) weak prior distributions were used for the remaining parameters of both models. Gibbs samples of 5000 iterations to burn in and 10000 in equilibrium were used for the Littlewood model.

Table I illustrates that the posterior mean number of bugs in the software initially is sensitive to the prior under both models. Sensitivity to the prior is not unexpected here as, for example, a Poisson(50) prior puts virtually no weight on values of N much above 100 and a Poisson(200) prior puts virtually all weight in the range (150, 250). This suggests that the we need to think very carefully about the prior settings when using either of these two models. Table I also illustrates that the JM model is somewhat more optimistic than the Litlewood model. Such results have been observed elsewhere, see e.g. (13).

Prior Mean	**Posterior Mean**	
	JM	**Littlewood**
50	91 (3)	113 (6)
100	104 (7)	147 (9)
200	194 (15)	230 (14)

Table I. Posterior mean (and standard deviation) estimates of N given different priors.

Assume that given the observed data, we now have to decide whether or not further testing is necessary. In Figure I, me illustrate how the optimal further test times for the JM and Littlewood models vary with different values for the parameter c_1 in the utility function in Equation 4. In all cases, we have assumed an initially optimistic prior for N, with mean 50, and the parameter T^{max} set at 500000 *cs*. The remaining utility function parameters are fixed at $c_2 = 0$ and $c_3 = 0.04$, respectively. Optimal test times were estimated by evaluating the expected utility over a grid of t values and choosing the test time in the grid with maximum utility.

The results shown by Figure 1 appear paradoxical at first. Littlewood's model predicts that a greater number of errors are left in the program than those predicted by the JM model, hence we would expect that further test times would be longer for Littlewood's than for JM's. Note however, that JM treats each fault in the code as being of the same size (ϕ), whereas Littlewood may be interpreted as treating the faults as being of variable sizes, see (12). Thus, under Littlewood, some of the faults assumed left in the program are likely to be very small and difficult to find, even in the maximum test period and further testing to find these faults will be unprofitable.

Clearly, the model used has a great influence on the optimal length of testing to be undertaken. This calls for using model uncertainty methods within decision making, see Draper (6) for some ideas.

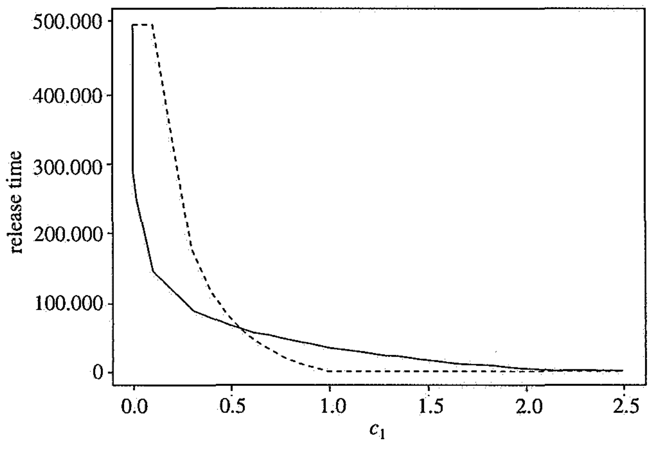

Figure 1. Optimal further test times under different values of c_1.

5. A MODIFICATION

In Section 3, we looked at estimating utility for possible test times T. We illustrate here how we can use a method of Bielza, Muller and Rios Insua (2), to estimate the optimal testing time T^{opt}. The approach described here is a hybrid algorithm; in (2) a Gibbs sampling approach is introduced, whereas in Virto, Rios Insua and Martín (2) a Metropolis approach is presented.

We use an augmented probability method, which, for the purposes of sampling, treats the utility function as a «probability». This is the reason why we ensured that the utility function defined in Section 3 was always positive. Briefly, we proceed as follows. Define $h(T, \theta, t_{m+1},..., t_N) \geq 0$ to be a probability density proportional to the integrand in (5). Then, we can simulate realisations from $h(\cdot)$ using the following algorithm.

1. Set initial values $\theta^{(0)}$, $T^{(0)}$, $i = 0$.

2. Generalte $\theta^{(i+1)}$ from $f(\theta|t_1,...,t_m)$.

3. Generate the remaining failures $t_{m+1},..., t_N$ from $f(t_m+1,..., t_N|\theta^{(i+1)}, t_1,..., t_m)$.

4. Generate a new test time T^{cand} from a probing distribution $g(T|T^{(i)})$.

5. Accept the candidate with probability

$$p = \min\left\{1, \frac{U\left(T^{cand}, \theta^{i+1}, \mathrm{t}\right) g\left(T^{(i)}\middle|T^{cand}\right)}{U\left(T^{(i)}, \theta^{i+1}, \mathrm{t}\right) g\left(T^{cand}\middle|T^{(i)}\right)}\right\}$$

and set $T^{(i+1)} = T^{cand}$. Otherwise set $T^{(i+1)} = T^{(i)}$.

6. $i = i + 1$, goto 2.

Given this algorithm, it can be shown (see[2]) that the mode of the sample of test times generated is the optimal test time T^{opt}. The mode may be estimated from the sample by inspection of a histogram of the test times generated.

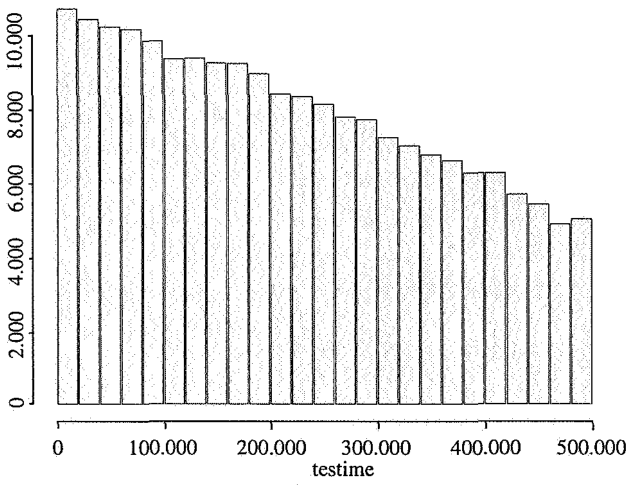

Figure 2. Histogram of test times.

Note that although we have outlined the algorithm for use after data $(t_1,...,t_m)$ have been observed, it could equally well be used to perform a full preposterior analysis, i.e. to estimate an optimal length of one-stage testing based on the original prior distribution alone.

We now use one of the situations given in the Example in order to illustrate the algorithm outlined above. Specifically, consider a situation where the optimal test time is 0, for example with utility parameters ($c_1 = 1$, $c_2 = 0$ and $c_3 = 0.04$). A histogram of the test times T generated by the algorithm is given in Figure 2.

We can see that the histogram is unimodal at 0, indicating that it is optimal to release the software immediately. This agrees with the results in Figure 2.

In this specific case, somewhat more iterations were needed to reach equilibrium than with the simpler Gibbs samplers described earlier. Bielza et al. (2) describe cases in which this novel approach may be more efficiente.

6. DISCUSSION

We have illustrated the use of modern Bayesian simulation methods to make inferences about the Littlewood and Jelinski Moranda models for software failure and to assess the optimal time for further software testing given these models. We have illustrated that inference for both models is somewhat sensitive to the prior and that the

Jelinski-Moranda model is somewhat more confident than the Littlewood model.

The issue of sensitivity is important and suggests that we might also consider hierarchical prior models to give less informative priors. One possibility is examined (for JM) by Rodrigues, (17).

We have also illustrated various approaches to solve the decision making issue of how much more testing time should be considered.

Many other software reliability models and utility functions could be considered within our framework. Moreover, various other decision making problems may be considered, such as multistage testing problems. These will the object of further work.

ACKNOWLEDGMENTS

This research has been supported by grants from CAM and CICYT.

REFERENCES

1. Abdel Ghaly, A.A. PhD Thesis, City University, London, 1986.
2. Bielza, C., Muller, P. & Ríos Insua, D., Monte Carlo methods for decision analysis with applications to influence diagrams, to appear in *Management Science,* 1999.
3. Chen, T.Y. & Yu, Y.T. On the expected number of failures detected by subdomain testing and random testing, *IEEE Trans. Soft. Eng.,* **22,** 109-119, 1996.
4. Dalal, S.R. & Mallows, C.L. When should one stop testing software? *J. Amer. Statist. Assoc.,* **83,** 872-879, 1986.
5. Dale, C.J. Software Reliability Evaluation Methods. *British Aerospace Dynamics Group,* **ST-26750,** 1982.
6. Draper, D. Assessment and propagation of model uncertainty, *J. Royal Stat. Society B,* **57,** 45-98, 1995.
7. Gilks, W.R., Richardson, S. & Spiegelhalter, D.J. Introducing Markov chain Monte Carlo. In *Markov Chain Monte Carlo in Practice,* eds. W.R. Gilks, S. Richardson & D.J. Spiegelhalter, 1-20, London, Chapman & Hall, 1996.
8. Hierons, R.M. & Wiper, M.P. Estimation of failure rate using random and partition testing, *J. Software Testing, Verification and Reliability,* **7,** 153-164, 1997.
9. Jelinski, Z. & Moranda, P. Software Reliability Research. In *Statistical Computer Performance Evaluation,* ed. W. Frieberger, 465-484, New York, Academic Press, 1973.
10. Kapur, P.K. & Garg, R.B. Cost-Reliability optimum release policies for a software system under penalty cost, *Int. J. Sys. Sci.,* **20,** 2547-2562, 1989.
11. Keiller, P.A., Littlewood, B., Miller, D.R. & Sofer, A. Comparison of software reliability predictions, *Digest FTCS,* **13,** 128-134, 1983.
12. Littlewood, B. Stochastic reliability growth: a model for fault removal in computer programs and hardware designs. *IEEE Trans. Reliab.,* **30,** 313-320, 1981.
13. Littlewood, B. Forecasting Software Reliability, Lecture Notes in Computer Sciencie No. 341, Berlin, Springer-Verlag, 1989.
14. Littlewood, B. & Sofer, A. A Bayesian modification to the Jelinski-Moranda software reliability model. CSR Technical report, City University, London, 1985.
15. Littlewood, B. & Wright, D. Some conservative stopping rules for the operational testing of safety critical software. *IEEE Trans. Soft. Reliab.,* **23,** 673-683, 1997.
16. Muller, P. Simulation Based Optimal Design. To appear in *Bayesian Statistics 6,* eds. J. Berger, J.M. Bernardo, A.P. Dawid & A.F.M. Smith, Oxford, Oxford University Press, 1998.
17. Rodrigues, J. Inference for the software reliability using asymmetric loss functions: a hierarchical Bayes approach. *Commun. Statisc. Theory Meth.,* **27,** 2165-2171, 1998.
18. Singpurwalla, N. Determining an optimal time for testing and debugging software. *IEEE Trans. Soft. Eng.,* **17,** 313-319, 1991.
19. Singpurwalla, N. & Wilson, S. Software Reliability Modeling. *Int. Stat. Rev.,* **62,** 289-317, 1994.
20. Virto, M.A., Ríos Insua, D., Martín, J. Markov Chain Monte Carlo methods for project management under uncertainty and resource constraints. *Tech, Rep. UPM,* 1997.
21. Yamada, S. & Osaki, S. Optimal Software Release policies with simultaneous cost and reliability requirements, *Europ. J. Oper. Res.,* **31,** 46-51, 1987.
22. Zeephongsehul, P. Stackelberg strategy solution for optimal software release policies, *J. Optimization Theory & Applications,* **91,** 215-233, 1996.