

Software reliability assessment using accelerated testing methods

Willa K. Ehrlich,

AT&T Network and Computing Services, Red Hill, USA

Vijayan N. Nair†

University of Michigan, Ann Arbor, USA

and M. S. Alam, W. H. Chen and Mark Engel

AT&T Network and Computing Services, Red Hill, USA

[Received June 1997. Revised September 1997]

Summary. The use of operational profiles and usage-based testing has received considerable attention recently in the software engineering literature. Testing under the actual operational profile can, however, be expensive, time consuming or even infeasible in situations where the performance of a system is dominated by infrequent but highly critical events. We consider a real application that deals with telecommunications network restoration after network failure caused by cuts in fibre optic cables. We use this application to demonstrate the usefulness of traditional accelerated testing methods to test and estimate software reliability. These methods, which have been extensively used in hardware reliability, have an important role to play in software reliability assessment as well.

Keywords: Accelerated testing; Poisson regression; Software reliability; Software testing; Telecommunications network restoration

1. Introduction

There has been considerable discussion about operational profiles and usage-based testing in the software engineering literature in recent years (see, for example, Musa (1993) and Walton *et al.* (1995)). Testing under the actual operational profile can, however, be expensive, time consuming or even infeasible in certain situations. This paper considers a telecommunications network restoration software system that is designed to perform alarm reporting and network restoration after failures caused by cuts in fibre optic cables. These are highly critical events, but they occur very infrequently — about 10 times a year on the AT&T network in the USA. Most of the time, the system is quiescent, processing routine background transactions. Thus, it is not practical or even feasible to do testing based exclusively on the operational profile. This paper describes an alternative approach for software testing and reliability assessment. A corner-stone of the approach is the use of traditional accelerated testing (AT) methods that have been used extensively in the area of hardware manufacturing.

In hardware reliability, the units are tested at accelerated stress conditions to induce failures, and the results are extrapolated, using accelerated failure time regression models, to

† *Address for correspondence:* Department of Statistics, 1429 Mason Hall, University of Michigan, Ann Arbor, MI 48109-1027, USA
E-mail: vnn@umich.edu

predict reliability in normal operating conditions (Nelson, 1990). The two basic means of acceleration are increasing the usage rate and increasing the aging rate. The latter can be accomplished by increasing the levels of stress factors such as temperature, power and humidity. AT is necessary as few units will fail under normal operating conditions in a reasonable period of test time. AT is recognized to be a useful tool for obtaining timely information about product reliability in hardware manufacturing.

This paper uses a real application dealing with telecommunications network restoration to demonstrate the usefulness of these methods for software testing and reliability estimation. Although the approach and techniques are discussed and illustrated in the context of the telecommunications network restoration software system, they should also be useful in other software applications. There are also other approaches in the literature for testing safety critical systems and systems that are intended to perform under emergency conditions. Interested readers are referred to Butler and Finelli (1993), Levenson (1991), Littlewood and Strigini (1993), Parnas *et al.* (1990) and references therein.

The paper is organized as follows. The background of the application is described in Section 2. The test design and the use of AT are described in Section 3. In our test design, we used the operational profile to identify the various states of the system when a network failure occurs. The performance of the software under these states was studied in a test environment that was designed to simulate the AT&T network. The system was operated under several levels of highly 'stressed' environments to induce failures. Sections 4 and 5 describe different approaches to modelling the observed failures and extrapolating the results to estimate reliability in normal use conditions. Section 4 discusses a purely empirical approach based on Poisson regression of the number of failures. A more elaborate analysis based on a model for the underlying software failure mechanism is discussed in Section 5. This is similar in spirit to recent work in hardware reliability where underlying subject-matter knowledge is used to develop better acceleration models (see Meeker and LuValle (1995)). Section 6 illustrates how these results can be used to estimate operational reliability. The paper concludes with some general remarks in Section 7.

2. The network restoration system

The introduction of fibre optic technology has greatly increased telecommunications network capacity in recent years. Unfortunately, this has also made the networks more vulnerable to failure from 'cable cuts' caused by events such as flooding and construction work. A single cable cut can affect hundreds of thousands of customers or circuits. For example, a fibre cut in the Newark, New Jersey, area in November 1988 caused the loss of over 270 DS3 paths — or over 180000 telephone circuits — and it took more than 15 h to re-establish service. (A DS3 is a digital transmission system designed to transmit signals at the rate of 45 Mbits per second.) Because of the potential for such a massive disruption of service, there is a need to develop methods that automatically detect network failures and restore service by rerouting the affected paths. The FASTARSM operations system platform is a set of systems that provide fast automated restoration capabilities for the AT&T network (see Chao *et al.* (1994)). The FASTAR technology, introduced in 1991, has significantly increased the reliability of the network and reduced the effect of cable cuts. For example, the record floods in 1993 washed away a major back-bone cable between Kansas City and St Louis, yet it took only a few minutes to restore the service to more than 250 DS3 paths.

We use the highly simplified network in Fig. 1, consisting of six central offices A, B, C, D, E and Z, to describe the restoration process. Central offices can be viewed as nodes in a

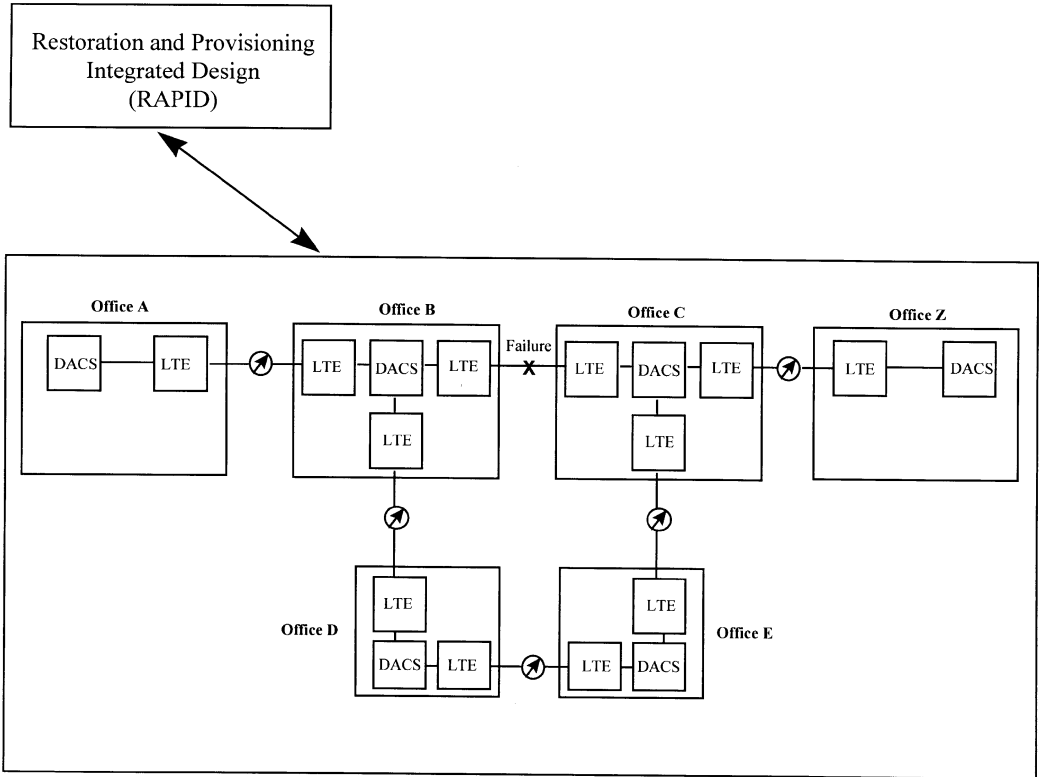



Fig. 1. Simplified view of the network with six central offices: , fibre optic cable

telecommunications network. The FASTAR system is a collection of computer systems that work together to manage the various activities involved in the restoration process (Chao *et al.*, 1994). The components of the FASTAR system that are relevant to our discussion are shown in Fig. 2. A central controller ('RAPID') and remote node controllers (RNCs), located within each central office, form the back-bone of this restoration system. The RNC interfaces with various devices in a central office, including the line terminating equipment (LTE), restoration test equipment (RTE) and digital access and cross-connect system (DACS).

The restoration process consists of the following steps.

- Gathering and reporting alarms*: if a cable cut occurred between offices B and C in Fig. 1, the LTE at these offices would report the loss-of-signal failure to an RNC. The RNC then uses a given protocol to determine whether the failure is real (rather than transient) and then reports it to the central controller (RAPID).
- Determining and implementing alternative paths*: the central controller waits for a predetermined time before compiling a list of paths that require restoration and initiating the restoration process. This consists of first dynamically computing the best alternative path that is available to reroute the calls. For example, the rerouted path may go through offices B, D, E and C in Fig. 1. The central controller then commands the DACSs at these offices to switch to the alternative path.
- Doing path assurance*: the central controller needs to assure that the alternative path

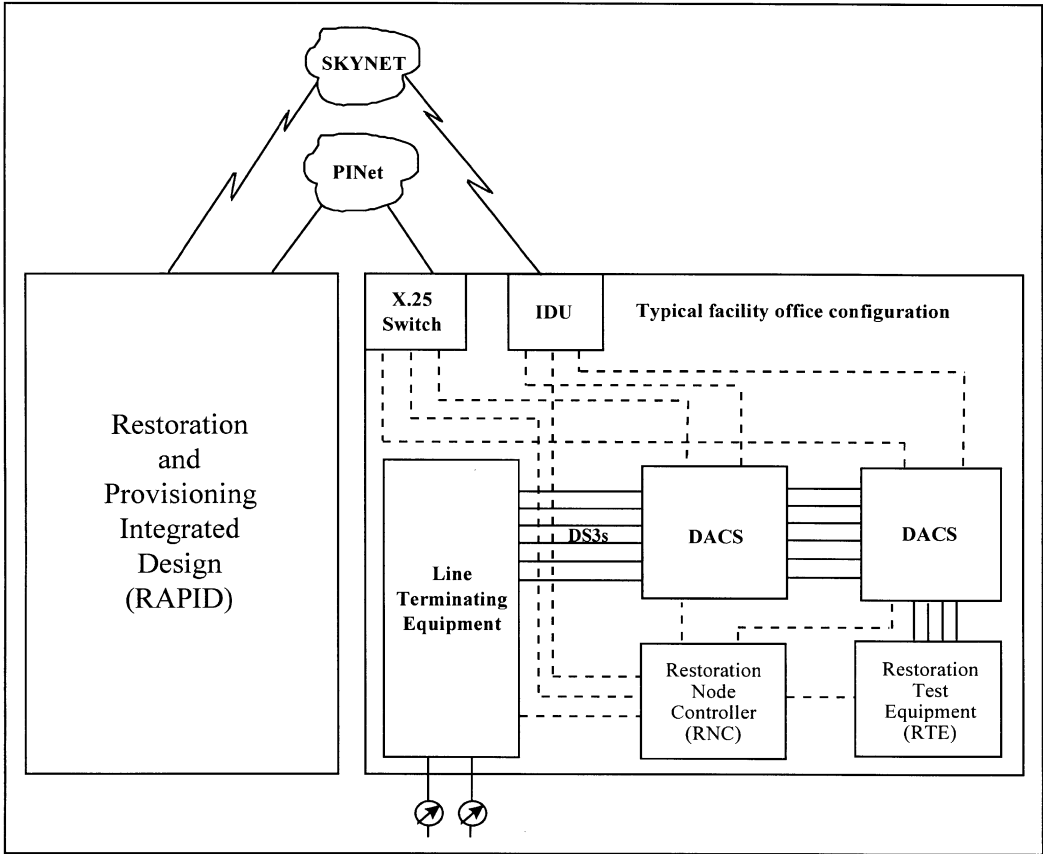


Fig. 2. Relevant components of the FASTAR system: IDU, indoor unit; —, lines; - - -, links; ⊗, fibre optic cable

has connectivity. To do this, it issues a service verification (SV) request to one of the RNCs. The RNC commands the DACSs to bridge the incoming signal to the RTE receiver port and then queries the status on that signal. Once the SV is successful, the path is considered restored. The SV request for rerouted paths can be assigned to RNCs that did not perform alarm, gathering and reporting (e.g. RNCs at offices A and Z in Fig. 1) or to RNCs that performed alarm gathering and reporting (i.e. RNCs at offices B and C in Fig. 1).

3. Test design

The interface between the RNC and the central controller was modified in a new release resulting in a major redesign of the RNC software. Thus, the new software had to be tested and its operational reliability assessed.

As noted earlier, the RNC system is mostly quiescent, involved in background processing. This includes activities such as processing maintenance requests, database transactions, operator-initiated commands, RNC-initiated periodic events and RNC responses to equipment and equipment link failures. Exceptional events such as cable cuts occur rarely.

When such an exceptional condition occurs, the RNC system can switch to one of several modes of processing ('operational states'). We used the operational profile to identify these various states. Our test strategy was to induce exceptional events, to execute the RNC in each of these states and to observe its performance. Operational reliability can then be calculated by weighting and averaging the performances according to the probabilities of occurrence of the various states as given by the operational profile (see Musa (1993)).

All the testing was done on a test environment that had been developed to simulate the AT&T network. This test environment handles all the inputs and outputs identified in the operational profile and has the ability to test the effect of exceptional condition scenarios such as cable cuts and network equipment failures. Simulator environments are designed with a programmable interface. This allows the development of automated test scripts to drive the test runs. Data from each test run can also be collected automatically via the programmable interface.

3.1. States of the system

The following were identified as the important states of the RNC during exceptional condition processing. In addition to these states, the RNC can also be quiescent, state Q, where it is involved in the background processing activities noted earlier. The RNC does these background processing activities even during the exceptional events.

- (a) State A represents background load processing in combination with SV activity related to cuts in the cables. In this state, the RNC merely does path assurance and SV requests for the paths that are to be restored. It does not do alarm generation and reporting associated with the cable-cut condition.
- (b) State B is background load processing in combination with alarm gathering and reporting activity related to cuts in the cables. In this mode, the RNC only reports failed paths and does not subsequently engage in SV and path assurance.
- (c) State C is a combination of states A and B. In this state, the RNC does background load processing in combination with alarm gathering and reporting related to cuts in the cables followed by SV and path assurance activity related to a cable cut.
- (d) State D is background load processing and restoration activities in response to a non-cable-cut condition. This refers to the loss of service due to events other than a cable cut such as equipment failure or human errors. The number of paths involved in this is small, so, in our test, we considered only the situation where the RNC performs both the alarm gathering and reporting as well as the path assurance components of a restoration.

3.2. Acceleration factor

It was expected that at least some of the inherent software failures in the system might be due to 'interactions' between background processing and transactions related to exceptional conditions. The usual rate of background processing was rather low, so there was a high probability that these failures will be undetected. So, the rate of background transactions must be increased to induce failures (similar to hardware reliability). However, we also needed to estimate the number of failures in normal operating conditions. Thus, the test was conducted at three different acceleration conditions that corresponded to 10, 100 and 200 times the actual rate of background processing in field conditions. The actual acceleration rates in the test turned out to be lower than these intended settings. An analysis of the actual rate of background transactions (after the test) showed that the actual acceleration rates were 10, 79 and 130. Nevertheless, we continue to use the labels acc10, acc100 and acc200 to refer

Table 1. Sequence of exceptional condition states applied to the RNC system test

ABCD	CDBA	BCAD	DACB
CDBA	ABDC	DACB	BCAD

to the acceleration settings. Whenever the actual values of the acceleration factors are used in the analysis, this is made clear.

3.3. Order of test runs

The tests were conducted by inducing a cable-cut exceptional event and executing the RNC system in one of its several possible states. Because of limited resources, all the states had to be executed on the same RNC test machine. The different states were executed sequentially with the ‘background load only’ state, state Q, run in between for 15 min to ‘renew’ or reset the system following exceptional conditions.

There was still the possibility of the effect of one state carrying over to and affecting the performance of the subsequent state because of the cumulative nature of system resource usage (i.e. central processor unit (CPU), disc and memory swapping) during the execution of a performance run. To account for such effects as well as other patterns or time trends (e.g. process size growing in time due to failure to deallocate memory previously allocated resulting in increasing performance degradation), we used a balanced design to determine the order of state execution. The particular design, obtained from two 4×4 Latin squares, is shown in Table 1. There are 32 runs in all, with each state being replicated eight times. Each pair of states occurs equally often, and we can use this to estimate any effect due to prior states. There were, however, no time trends or carry-over effects, so this aspect of the problem is not discussed further.

The test was run as follows. For each background load acceleration, we tested all 32 exceptional condition states in a single run lasting for approximately 30 h. At the end of the 32 runs, the system was run for an extended period of time under low level background load to reinitialize the system. Then, the same sequence of 32 exceptional condition states was rerun under the new background load acceleration setting.

4. Empirical modelling of the number of failures

Table 2 shows, for each state, the number of runs (out of eight) in which there was at least

Table 2. Number of runs (out of eight) in which at least one failure occurred

State	Number of runs for the following acceleration settings:		
	10	100	200
A	3	8	8
B	0	0	0
C	1	8	8
D	0	0	0
Q	0	0	0

one failure. There were no failures during any of the eight runs for states B, D and Q. Thus, these states are not considered further in subsequent analyses. For state A, there was at least one failure in three of the eight times under acc10 and in all eight of the eight times under acc100 and acc200. For state C, there was at least one failure in one of the eight times under acc10 and in all eight of the eight times under acc100 and acc200.

A root cause analysis revealed that all the failures were attributable to the same fault: SV requests being timed out before being processed by the RNC. The nature of this fault is discussed in more detail in the next section. This failure indicates that some paths may not be restored, so it is important to estimate the number of paths that are not restored because of this failure under the usual no-acceleration condition. We discuss first a simple approach based on modelling the expected number of failures as a function of the acceleration condition and extrapolating it to normal operating conditions.

The actual numbers of failures (number of SVs that were timed out) under each execution of states A and C at the various acceleration settings are given in Table 3. We modelled these failures as Poisson random variables and used Poisson regression analyses to determine whether and how the parameters varied with

- (a) different acceleration settings,
- (b) different states of the RNC,
- (c) carry-over effects due to prior states and
- (d) CPU usage.

A log-link function was used for the Poisson parameter, and in these preliminary analyses we treated acceleration settings as a nominal variable. The results indicated that only the acceleration effect was important, with the expected number of failures increasing as the degree of background load acceleration increased. None of the other effects, including the system's current state (A or C), prior state and CPU usage, was found to be important.

To extrapolate to estimate the expected number of failures at the normal background

Table 3. Number of failures under states A and C at the various acceleration settings

Run	State	Numbers of failures for the following acceleration settings:		
		10	100	200
1	A	2	6	6
4	C	0	3	4
5	C	0	6	3
8	A	0	6	5
10	C	0	4	3
11	A	3	4	3
14	A	0	4	6
15	C	0	5	6
17	C	0	7	5
20	A	0	5	5
21	A	3	4	6
24	C	0	5	8
26	A	0	2	6
27	C	5	2	7
30	C	0	3	4
31	A	0	3	7

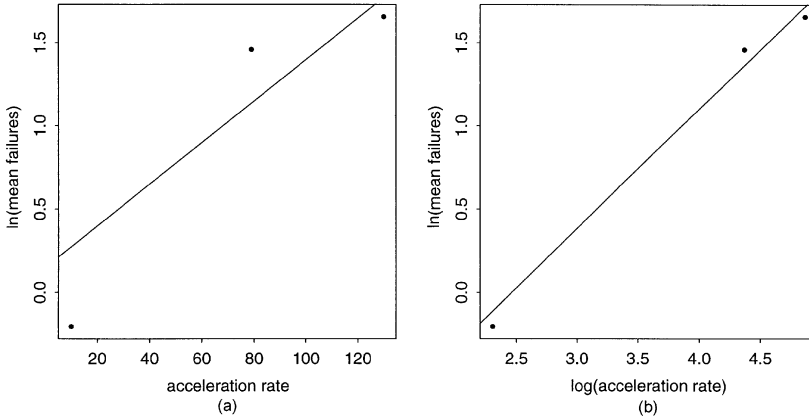


Fig. 3. Plots of $\ln(\text{mean failures})$ against (a) acceleration rate and (b) $\ln(\text{acceleration rate})$: —, fitted models from Poisson regression analyses

processing rate, we need a regression model for the Poisson parameter $\lambda(x)$ as a function of the acceleration settings. We considered the regression model

$$\ln\{\lambda(x)\} = \beta_0 + \beta_1 x, \quad (1)$$

where $\lambda(x)$ is the Poisson parameter and $x = g(\text{acceleration rate})$ for some function $g(\cdot)$. In these analyses, we used the actual acceleration rates of 10, 79 and 130. Fig. 3 shows the relationship between $\ln(\text{mean failures})$ and the acceleration rates for two models: $x = \text{acceleration rate}$ and $x = \ln(\text{acceleration rate})$. The lines correspond to the fitted model from a Poisson regression analysis. The log-linear relationship provides a better fit, and this was also confirmed from a comparison of the residual deviances. The half-normal plots of the residual deviances were reasonably linear for both models.

We can now use the regression model to extrapolate and obtain the estimated mean number of failures at the no-acceleration setting. For the log-linear relationship with $x = \ln(\text{acceleration rate})$, the estimated mean number of failures is 0.17 with a standard error of 0.09. If we used the model with a linear relationship $x = \text{acceleration rate}$, the estimated value is 1.18 failures with a standard error of 0.16. The two models give slightly different estimates. On the basis of the data alone, we might choose the model with a log-linear relationship. However, as there are only three acceleration settings, we need to be cautious in relying exclusively on empirical models.

The dangers of extrapolation are well known, yet extrapolation is a necessary evil in AT. There are many examples in the reliability literature where it is difficult to determine which regression model is reasonable on the basis of the data, and different models lead to drastically different estimates of reliability. Because of this, there has been more emphasis recently in the hardware area on using the underlying subject-matter knowledge to develop more reliable acceleration models and to do extrapolation. For example, Meeker and LuValle (1995) showed the inadequacy of the usual accelerated failure time models for a particular application and used the underlying subject-matter knowledge to develop a reliability kinetic model for the acceleration transform. In an analogous manner, we use knowledge of the underlying software failure process to obtain more reliable extrapolation in the next section.

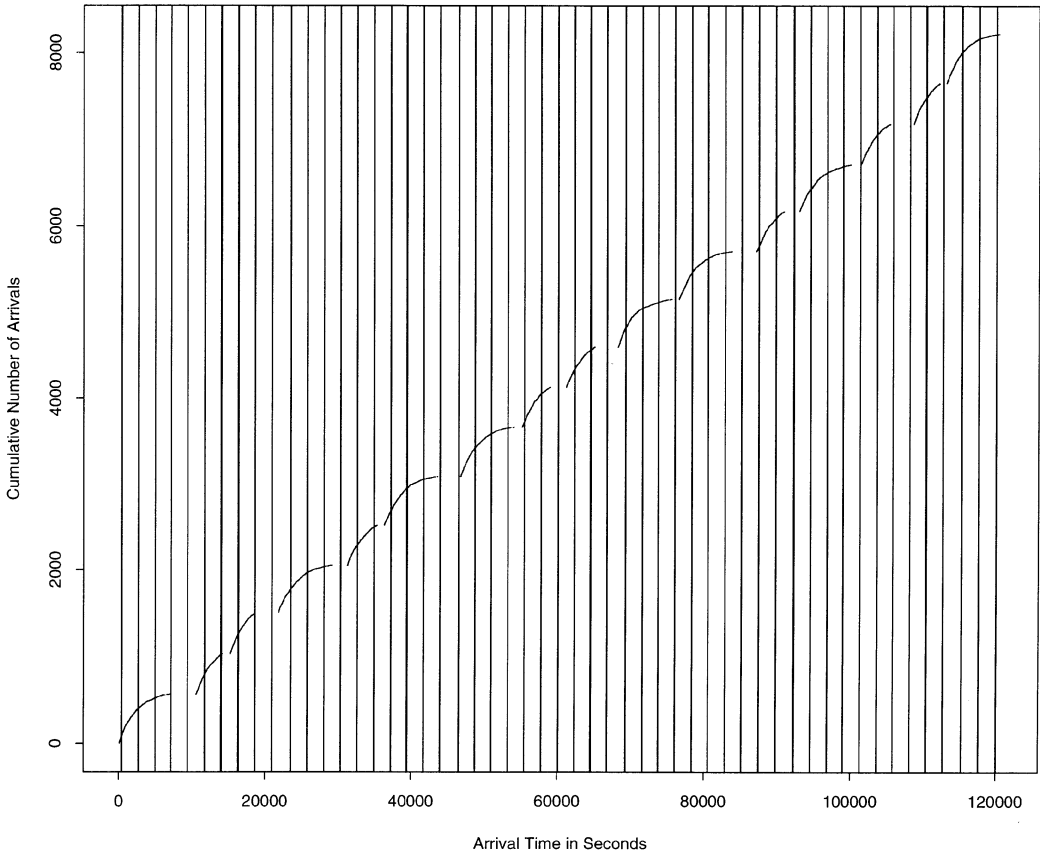


Fig. 4. Arrival patterns of SV requests and dbaudits for setting acc100

5. Modelling the software failure process

5.1. The anatomy of a software failure

As indicated previously, all the software failures that occurred during the RNC system test were caused by the same underlying fault. This fault occurred when an SV request arrived at the RNC while a *dbaudit* (database audit) transaction was already being processed. Under certain conditions, which we now elaborate on, the SV request was timed out and the path was not restored, leading to a software failure. These failures were observed only for states A and C; no SV request was generated under state B and the number of SV requests under state D was too small to observe this failure.

Fig. 4 shows the sequence of the SV request arrivals and the dbaudit transactions at the RNC system under acceleration setting acc100. The various curves correspond to the cumulative number of SV requests under states A and C. See Table 1 for the order of the runs A and C. The vertical lines correspond to the dbaudit transactions.

Fig. 5 provides a magnified view of Fig. 4 with the area between the vertical lines corresponding to the duration of a single dbaudit transaction. As can be seen, several SV requests can arrive at the RNC during the period of a dbaudit transaction. Some of these requests may not be processed, leading to a failure. A root cause analysis of this fault revealed the following.

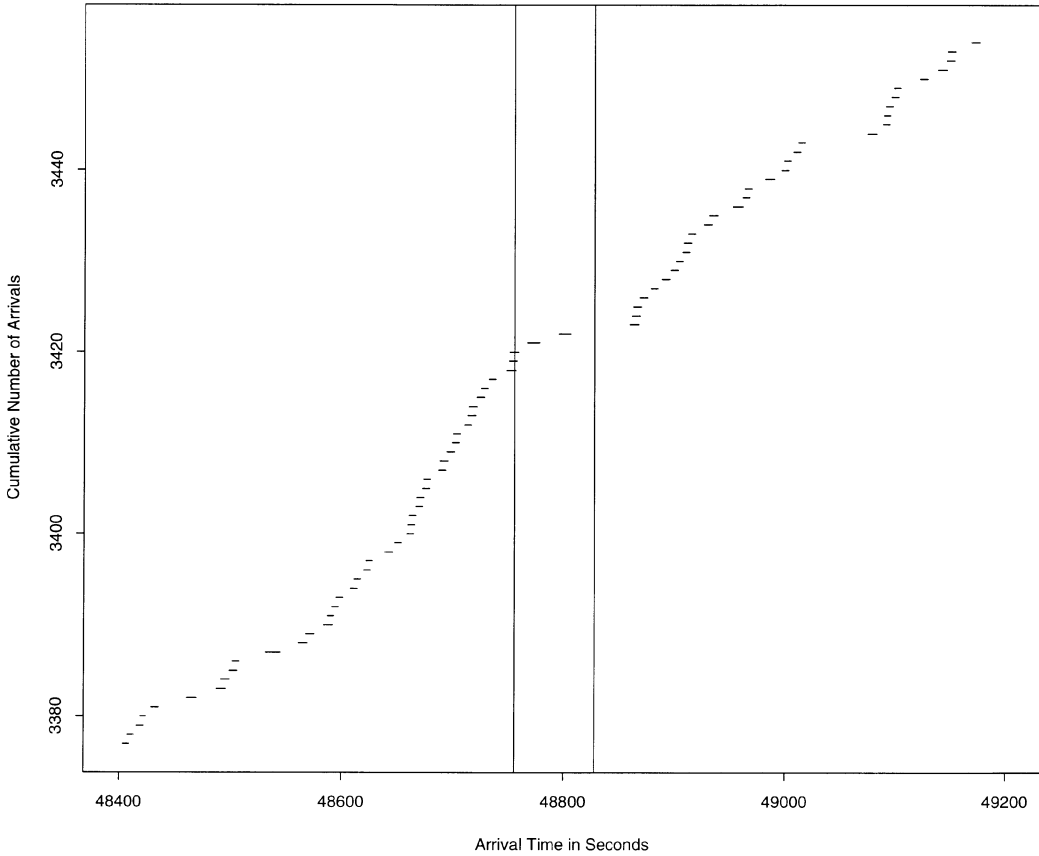


Fig. 5. Magnified view of the arrival patterns of SV requests and dbaudits for setting acc100

The RNC software consisted of various servers which process the dbaudit and SV request transactions. One of them was the database daemon (DBD). If a dbaudit operation was currently being serviced by the DBD and an SV request arrived at the DBD, it was placed on the DBD queue. The dbaudit transaction required a relatively long DBD service time, and so all SV requests arriving at the DBD server would queue in the DBD queue. There was a threshold setting for the time-out parameter, and if these SV requests remained on the DBD queue longer than the threshold period they were timed out, and the restoration process for the corresponding path was not completed.

We might conclude from this that, if the SV arrived at the RNC a fixed time interval after the dbaudit transaction, it will be timed out. However, the situation was more complicated because the transactions must access several servers, and an SV request that arrived at the RNC just after a dbaudit arrival can reach the DBD server before the dbaudit transaction and hence be served. The data showed that the time-out mechanism depended not only on the absolute offset of the SV request arrival time with respect to the dbaudit transaction arrival time but also on other factors such as how long the dbaudit transaction was resident on the RNC. If A denotes the dbaudit arrival time and R denotes its residence or transaction time, then all SVs arriving in the period

$$[A + 0.05R, A + 0.85R] \quad (2)$$

were timed out in our study. Thus, we used this as the failure interval or window in our analysis.

Using the interval in expression (2), we can now estimate the expected number of failures at the normal background load (no acceleration) as the number of SVs that are expected to arrive in this window. To do this, we need to develop a model of the SV arrival process as well as that of the dbaudit transaction process under no background acceleration.

5.2. The service verification arrival process

The number of SV requests arriving at an RNC during an actual cable cut in the field depends on several factors, including the number of paths that are affected. Further, an SV request can be assigned by the central controller to one of several RNCs. Thus, the actual number of SV requests arriving at an RNC is random, and it may be reasonably modelled as a Poisson random variable with parameter θ which is allowed to vary across the RNCs, depending on the number of paths that they can serve. Further, the nature of the restoration process and field data suggest that the numbers of SV requests arriving in mutually exclusive time periods are likely to be independent of each other. This, together with the arrival patterns in Figs 4 and 5, suggests a non-homogeneous Poisson process as a model for the SV arrival process. Let $\Lambda(t)$ be the cumulative intensity function of the non-homogeneous Poisson process. For this application, $\Lambda(t)$ should eventually reach a finite asymptote with $\Lambda(\infty) = \theta$, the mean of the total number of SV requests.

In our accelerated test, the total number of paths affected by the cable cut (and hence the number that needed to be restored) was fixed at (approximately) 570. The actual number varied slightly from run to run. The test under state A was run until the restoration process was (supposedly) complete, i.e. all the SV requests had arrived. For state C which involved both alarm gathering and reporting as well as path assurance, the test was designed to stop as soon as a fixed number of paths, 467, were restored. This corresponds to type II censoring. From the well-known property of a non-homogeneous Poisson process, the SV arrival data (cumulative times of arrival) under state A, conditional on the total number of arrivals, can then be viewed as order statistics from a distribution with cumulative distribution function (CDF) $F(t) = \Lambda(t)/\Lambda(\infty)$ (see Cox and Lewis (1966), pages 27 and 29). For state C, the data can be viewed as type II censored data from $F(t)$.

Among the various commonly used distributions, the two-parameter Weibull distribution

$$F(t) = 1 - \exp\{-(t/\alpha)^\beta\}$$

provided the best fit for the data. Fig. 6 shows the Weibull probability plots of the data under states A and C for a typical run. Fig. 6(a) is the probability plot of the first execution of state A under setting acc100. It looks reasonably linear except perhaps in small regions in the lower and upper tails. Fig. 6(b) is the probability plot of the data from the first execution of state C under setting acc100. This plot confirms that the Weibull distribution is a reasonable model.

We first fitted the Weibull parameters separately for states A and C. Although these values were quite close to each other, the differences were statistically significant. The statistical significance was mostly due to the fact that the sample sizes were large. As there was no plausible reason for the difference and the values were fairly close to each other, we decided to combine data from both states and to fit a common distribution. The fitted values of this

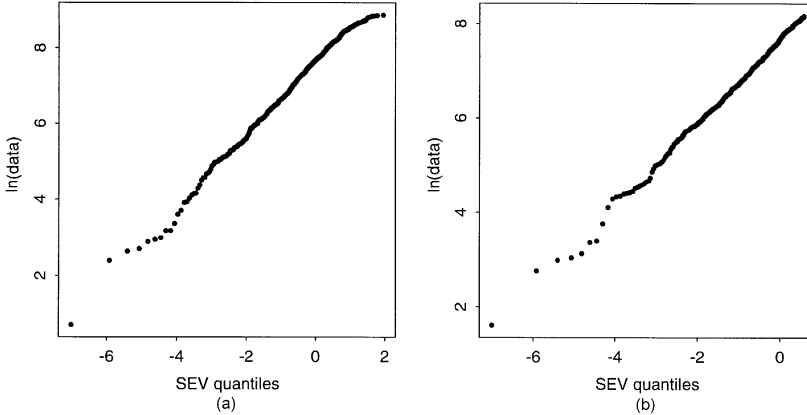


Fig. 6. Weibull probability plots of the data under setting acc100 for the first run for (a) state A and (b) state B

common Weibull model were $\hat{\alpha} = 1808$ with a standard error of 14.5 and $\hat{\beta} = 1.16$ with a standard error of 0.006.

In summary, our model states that the cumulative intensity function of the SV arrival process is proportional to this estimated Weibull CDF. It can be shown that the corresponding intensity function (given by the Weibull density function) has a unique mode at about 182 s, so the intensity of the SV arrivals peaks at about 182 s into the process and then decreases to 0.

5.3. Database audit transactions

Field data indicated that the dbaudit transactions arrive at the rate of one every 50 h in actual field conditions. Further, the arrivals are fairly regular, so for practical purposes we can take the arrival times to be fixed at one every 50 h.

However, there was not much information from the field data about dbaudit transaction residence times, so we analysed the test data and extrapolated the results to estimate the residence times under field conditions (no acceleration). Fig. 7 is the box plot of the dbaudit residence times at the three different acceleration settings. This suggests that the dbaudit residence time decreased with background load acceleration. This somewhat counter-intuitive behaviour can be explained by the dbaudit's use of 'caching' during the test—data from the database must first be brought from disc into memory, so the initial invocation of a dbaudit requires extra service time and hence extra residence time. Subsequent invocations of a dbaudit require less servicing because in-core database data can be utilized. Increased settings for background load acceleration result in more dbaudit invocations, which explains why dbaudit residence times decreased with background load acceleration.

We fitted an accelerated failure time regression model (Nelson, 1990) to the residence times at the three different settings and extrapolated to obtain the residence time at the no-acceleration setting. Among the commonly used models, the log-normal distribution provided the best fit. Again, we considered several functions of the acceleration settings as the regressor variable. Among these, the model with $x = \text{acceleration}$ fitted the data best. This yielded a median dbaudit residence time under an unaccelerated background load of 66.5 s with a standard error of about 3.8 s. This value was close to the median of the residence times observed under setting acc10. This gave us confidence in the extrapolation as there were only

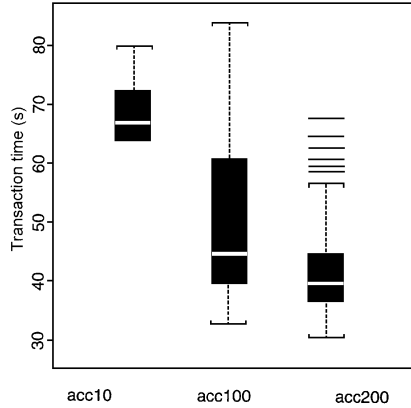


Fig. 7. Box plots of dbaudit transaction times under the various acceleration settings

five dbaudit transactions under setting acc10 and the effect of caching should thus be minimal at this setting.

5.4. Estimating the number of service verification time-outs

We now estimate the expected number of failures under normal background operations (no acceleration) for our test design. The calculations discussed here are primarily of interest for comparison with the empirical extrapolation results of Section 4. A more general discussion of operational field reliability is deferred to Section 6.

Suppose that, as in our test design, we observed a cable-cut exceptional event that affected 570 paths. (This is more than twice the average number of paths affected in a cable cut in the field.) The expected number of paths not restored because of SV time-outs is then the expected number of SV requests times the probability that an SV request will be timed out under no background load acceleration. On the basis of our model, the probability that the SV request will be timed out is the expected value of

$$F(A + 0.85R) - F(A + 0.05R) \tag{3}$$

where F is the estimated Weibull CDF of the SV arrival process, A is the arrival time of the next dbaudit transaction relative to a given SV arrival process and R is the random residence time. The expectation here is taken with respect to the distributions of A and R .

As noted earlier, field data suggested that dbaudit transactions arrived regularly at the rate of one per 50 h under normal (unaccelerated) conditions. Since the onset of a cable cut, and hence the SV arrival process, is random, the distribution of A , the arrival time of the next dbaudit transaction relative to a given SV arrival process, can be taken to be uniform on the interval $[0, 180000]$ s (or 50 h). From the results in Section 5.3, the distribution of R , the dbaudit residence times, under no background load acceleration can be approximated by a log-normal distribution with $\hat{\mu} = 4.20$ and $\hat{\sigma} = 0.21$. Thus, the probability in expression (3) can be written more explicitly as

$$\int_0^T \int_0^\infty \left(\left[\exp \left\{ - \left(\frac{\alpha + 0.05r}{1808} \right)^{1.16} \right\} - \exp \left\{ - \left(\frac{\alpha + 0.85r}{1808} \right)^{1.16} \right\} \right] / 18000 \right) g(r) dr da \tag{4}$$

where $T = 18\,000$ s, the exponential terms are the probability of an SV being timed out in the interval $[\alpha + 0.05r, \alpha + 0.85r]$ under the fitted Weibull distribution and $g(r)$ is the estimated log-normal density for the residence times.

Using a numerical approximation, we computed the probability in expression (4) to be 2.55×10^{-4} . Since expression (4) is a highly non-linear function of the estimated parameter values, the usual techniques cannot be used to obtain standard errors or confidence intervals. We used a parametric bootstrap procedure (Efron and Tibshirani, 1993) to obtain an approximate 90% confidence interval as $(2.33 \times 10^{-4}, 2.79 \times 10^{-4})$. From these results, we estimated the expected number of SV failures for 570 cable cuts as 0.145 and the corresponding 90% interval to be (0.133, 0.159). It is interesting to observe that this estimated value is close to the value obtained from the log-linear regression model in Section 4.

To compute the expected number of failures under the test scenario of state C, we must take into account the fact that the test was stopped as soon as 467 of the 570 SV requests were generated. This is type II censoring with a censoring proportion of about 18%. We can approximate this by the corresponding type I censoring plan where the test is stopped after approximately 2981 s (which is the $1.0 - 0.18 = 0.82$ quantile of our fitted Weibull model). In other words, no failures can be observed if the failure window in expression (2) occurs after the first 2981 s of the SV arrival process. So, the probability of failure is now given by equation (4) with $T = 2981$ s. Again using a numerical approximation, we computed this probability to be 2.17×10^{-4} . From this, we estimated the expected number of failures as 0.124 with a 90% confidence interval of (0.114, 0.136). This is to be compared with the value of 0.17 obtained from the Poisson regression model in Section 4. On the basis of the empirical analysis there, we did not find a significant difference between the mean number of failures in states A and C. The more detailed analysis here, which exploits subject-matter knowledge of the underlying failure process and the test design, provides more reliable estimates, although in this example the values turned out to be about the same.

Instead of the expected number of failures, we might be interested in a worst case scenario. This occurs when a dbaudit transaction arrives during the period of most SV arrivals, i.e. when the failure window in expression (2) coincides with the period of the highest intensity of SV arrivals. We saw in Section 5.2 that the SV arrival intensity function is unimodal with a peak at about 182 s into the arrival process. From this, we estimated the worst case number of SV failures to be 11.3 out of a total number of 570 paths that are to be restored.

5.5. Performance analysis

To confirm these results and also to understand better the nature of the software fault, we developed a process flow model of the various transactions and servers in the RNC software system. Using this process flow model and the SES/Workbench modelling tool, we conducted a simulation study to estimate the number of SV time-out failures under the condition of no background acceleration. We do not go into the details of the model or simulation study here as these are not relevant to the scope of this paper. We note only that the results from this study were very close to the estimated number of failures in Section 5.4.

A natural question that arose from this study was whether the time-out threshold parameter in the software should be modified to reduce this failure rate or even to remove it altogether. This modification would affect the performance of the software in other areas, so such issues had to be considered carefully. One of the benefits of the performance analysis model that we developed was that it can also be used to do a cost–benefit analysis of such modifications.

6. Operational reliability

Software reliability is defined as the probability of failure-free operation over a specified time in a particular execution environment. The computation of the operational reliability for the AT&T network environment requires information about the network, such as the network architecture, rerouting protocols, the frequency of cable cuts and the distribution of the number of paths affected. Much of this information is proprietary, so we restrict attention here to an illustrative discussion. Consider, for example, the cable cut mentioned in Section 2 from the record floods in 1993. This involved 250 paths, which is slightly more than the number of paths affected in a typical cable cut. We can use the results in Section 5 to calculate the software reliability under such an exceptional condition. Then, software reliability is the probability that all 250 paths are restored without any failures (SV time-outs). As the failure interval in expression (2) is random, we must compute the probability of no failures over all possible failure intervals and average with respect to the distributions of the arrival and residence times of the dbaudit transactions. The probability of a failure in any fixed interval (a, b) is binomial with $n = 250$ and ‘success’ probability $p = F(b) - F(a)$ where $F(\cdot)$ is the fitted Weibull distribution. This follows from the well-known property that a non-homogeneous Poisson process, conditional on the total number of events, behaves like a multinomial process (Cox and Lewis, 1966). So the probability that there are no failures in the fixed interval (a, b) is $(1 - p)^{250}$. From all of this, we obtain an explicit expression for an estimate of the software reliability as

$$\int_0^T \int_0^\infty \left(\left[1 - \exp \left\{ - \left(\frac{\alpha + 0.05r}{1808} \right)^{1.16} \right\} + \exp \left\{ - \left(\frac{\alpha + 0.85r}{1808} \right)^{1.16} \right\} \right]^{250} / T \right) g(r) dr da, \quad (5)$$

where $T = 18\,000$ s. Again using a numerical approximation, we computed this value to be 0.977. The corresponding value for a cable cut of 570 paths is 0.971. These values are not as close to 1 as we would like. In fact, if there are 10 cable cuts of size 250 each in a year, then the probability of no failures in all of them is only $0.977^{10} = 0.79$. However, the expected number of paths affected by SV time-out failures is small: 0.06 for a single cable cut of 250 paths or 0.16 for 10 cable cuts of 250 each. This was deemed to be sufficiently small that the time-out threshold parameter in the RNC software was left unchanged.

It is a common practice in the software area to measure ‘time’ in CPU units instead of in calendar time. A resulting measure of reliability is field failure intensity as measured by the number of failures per CPU unit. In our study, we also collected data on CPU usage expended under the various states of the RNC. The CPU usage under state A was larger than that under state C, but this was due to the fact that there were fewer SV requests under state C. After adjusting for this, we found no difference in CPU usage. The estimated number of CPU units was 0.19. On this basis, the predicted rates of failures per CPU unit were 0.74 for states A and C and 0 for the other states. We can now use information from field data on the probability of the RNC being under the different states and weight these failure rates accordingly to obtain an overall field failure intensity.

7. Concluding remarks

We have used a real application to demonstrate the usefulness of some statistical methods for software testing and reliability estimation. This application dealt with situations where system performance is dominated by infrequent but highly critical events. An important part of our approach is the use of AT to induce software failures and to do extrapolations to

estimate reliability under normal operating conditions. These methods, which have been used extensively in hardware manufacturing, have an important role to play in software reliability assessment as well. We have also demonstrated the importance of using the underlying subject-matter knowledge about the failure process to do extrapolation. This is again consistent with recent directions in AT in hardware reliability.

Although the techniques have been presented and discussed in the context of a particular application, they are likely to be useful in other applications that also involve infrequent but highly critical events. They are being used to test the next generation (optical fibre) AT&T network that will support synchronous optical network technology, signal multiplexing and increased network intelligence.

Acknowledgements

We thank David James and Lian Xu for assistance with the data analyses, the Joint Editor and two referees for helpful comments, and Ron Brown and Edmund Gong for assistance in deriving the SV and dbaudit process flow models. We also thank John Musa and Raj Warty for introducing us to this problem and John Stampfel and Keith Lee for several insightful discussions that were relevant to this work. V. N. Nair's research was supported in part by National Science Foundation grants DMS-9404300 and DMI-9501217.

References

- Butler, R. W. and Finelli, G. B. (1993) The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Trans. Softwr. Engng.*, **19**, 3–12.
- Chao, C. W., Fuoco, G. and Kropfl, D. (1994) FASTARSM Platform gives the network a competitive edge. *AT&T Tech. J.*, July–Aug., 69–81.
- Cox, D. R. and Lewis, P. A. (1966) *The Statistical Analysis of Series of Events*. New York: Halstead.
- Efron, B. and Tibshirani, R. J. (1993) *An Introduction to the Bootstrap*. New York: Chapman and Hall.
- Folkesson, A. (1991) Reliability and maintenance technology. In *Operational Reliability and Systematic Maintenance* (eds K. Holmberg and A. Folkesson), pp. 1–8. New York: Elsevier Science.
- Levenson, N. G. (1991) Software safety in embedded computer systems. *Commun ACM*, **34**, no. 2, 34–46.
- Littlewood, B. and Strigini, L. (1993) Validation of ultrahigh dependability for software-base systems. *Commun ACM*, **36**, no. 11, 69–80.
- Meeker, W. Q. and LuValle, M. J. (1995) An accelerated life test model based on reliability kinetics. *Technometrics*, **36**, 133–146.
- Musa, J. D. (1993) Operational profiles in software reliability engineering. *IEEE Softwr.*, **10**, no. 2, 14–32.
- Nelson, W. (1990) *Accelerated Testing: Statistical Methods, Test Plans, and Data Analyses*. New York: Wiley.
- Parnas, D. L., van Schouwen, J. and Kwan, S. P. (1990) Evaluation of safety critical software. *Commun ACM*, **33**, no. 6, 636–648.
- Walton, G. H., Poore, J. H. and Trammel, C. J. (1995) Statistical testing of software based on a usage model. *Softwr. Prac. Exper.*, **25**, 97–108.