Statistical Certification of Software Systems

Alessandro Di Bucchianico, Jan Friso Groote,
Kees van Hee and Ronald Kruidhof [*][†]

Abstract

Common software release procedures based on statistical techniques try to optimise the trade-off between further testing costs and costs due to remaining errors. We propose new software release procedures where the aim is to certify that the software does not contain errors. The underlying model is a new discrete-time model similar to the Jelinski-Moranda model. The decisions are based on a mix of classical and Bayesian approaches to sequential testing and do not require any assumption on the initial number of errors.

keywords. Software release, error-free software, software testing, Bayesian statistics, stopping time, sequential testing, martingale, optional stopping.

1 Introduction

Errors in software systems are created during the development of the system. Such errors are often called *logical* errors. If a software system is error free, it will stay so for ever, unless someone is changing the software. This is a huge difference with physical systems that can have, besides logical errors, *physical* errors that occur during their lifetime due to some physical phenomenon like wear and tear. Since software systems may only have logical errors, it is a challenge to make software error free. Software systems are becoming more and more complex, and the number of possible states it can reach and the number of different action sequences it can execute, is astronomically large or even (theoretically) infinite. This means that it is practically impossible to make error free software systems and it is also (practically) impossible to test all possible states or action sequences. Nevertheless software systems are produced for clients and they require them to be error free. So software producers use release procedures based on a number of error free test runs, or on the time the system ran error free. To day release procedures are based on heuristics and best practices. It is important to have statistically based release procedures. There is an extensive literature on this topic; some recent papers are [1, 2, 3, 4, 5]. Most of these proposed methods consider the cost of repair and the cost of testing, and they use these costs to develop some cost optimal procedure. Another approach is to estimate the number of remaining errors and release software if this

---

[*] All authors are with LaQuSo, the Laboratory for Quality Software of the Eindhoven University of Technology. Alessandro Di Bucchianico is also with EURANDOM (Eindhoven) and Jan Friso Groote is also with CWI (Amsterdam).

[†] Email: {a.d.bucchianico,j.f.groote,k.m.v.hee}@tue.nl, r.kruidhof@student.tue.nl.

expected number is acceptable. We try to solve the "real" problem of the software industry. The software producer has to say to its client: To my best knowledge the system is error free. What is the probability that the claim of the software producer is wrong? Using the "common knowledge" that large software systems always have somewhere an error, we can say that this probability is one. However if we do not base the statement on common knowledge, but only on testing evidence, the situation becomes different. In fact we have a classical statistical problem. There is a null-hypothesis H_0 that says the software system contains one or more errors and there is an alternative hypothesis H_1 that says the software system is error free. We will use the test data to reject the H_0 and so to accept the H_1, when a sequence of error free tests is observed. If we discover an error we will repair it immediately and we start testing again. The type I error is the probability that we reject H_0, so we conclude that the system is error free, although there are errors. This probability should be acceptable, say 95%. The type II error the probability that we do not reject H_0 because we found an error, while H_1 is true. Clearly this probability is zero, so the power of the test equals one.

In this way we can use a well-known and generally accepted statistical procedure to construct a release procedure in a scientific way. Using this method does not include that the software producer will say to the client "the software is error free", but "the software passed the test with a confidence level of 95%". This problem seems not to have been studied in the software reliability, although a similar context appears in [6]. However, problems related to our problem also appear in other fields like proofreading (see *e.g.*, [7]).

In this paper we consider software as a black box. We do not give definitions of what an error is or what a test run is. In practice there are many types of errors, some are very serious and destroy important data, others make that we have to reset the system and again others make that we have to take some redundant actions which only take time and give us annoyance. The notion of a test run is not elaborated as well. We assume that we can perform test runs and that we may observe that they are either successful or a failure. We also assume detected errors can be repaired without introduction of new errors.

This paper is organized as follows. In Section 2 we describe our model and define the common elements of our release procedures. An important feature is that, contrary to many other approaches, we do not assume any knowledge on the initial numbers of errors in the software. The basic decision criterium is the number of error-free tests since the last error. If this number exceeds a certain threshold, then we declare the software to be error-free. By carefully choosing the threshold, we are able to indicate the probability of a premature release of the software system. Our release procedure in case we exactly know θ, the detection probability of errors is studied in Section 3. We show how to implement our procedure and study its performance in terms of the average number of required tests. Only if we have observed many similar systems during testing, we may have an estimate of θ. Often such an estimate is not available. Therefore we study in Section 4 a generalization of this model where we assume in a Bayesian manner that θ is unknown but has a prior distribution. This model is studied for uniform and Beta distributed prior distributions. We study a one-stage testing procedure with unknown detection probability in Section 5. The motivation for this one-stage testing procedure is to perform testing by a certification agency when their is already a test history from the producer of the software system. A fully adaptive setting where the testing parameters are updated after each detected error as if we were performing a one-stage testing, is studied through simulation. Surprisingly and fortunately, this procedure has premature release probability that is not much higher than the one-stage testing case.

2 The model and the release procedure

We assume that the number n of errors in a software system is unknown. The software system is subjected to a series of tests. A test is a walk through the state space of the system. Each walk has a finite length. A walk ends either in a bad state (*i.e.*, an error has been found) or in a good state (*i.e.*, a final state has been reached without detecting any error). We assume that the walks have been designed in such a way that errors are detected independently of each other and that each undetected error has a probability θ to be on the walk. So each error has probability θ to be detected during a test. This is not completely realistic, but may yield a reasonable approximation. When an error has been found, we repair it before we start testing again. Hence, in a test at most one error will be found. When there are r remaining errors , the probability of not detecting any error in one test equals $(1-\theta)^r$.

variable	page	definition
n	3	initial number of errors in system
θ	3	detection probability
p_i	3	success parameter of $X_i = 1-(1-\theta)^{n-i+1}$.
φ	3	passage probability = probability of non-detection = $1-\theta$
X_i	3	number of tests to find ith error after $(i-1)$th detected error
T_i	3	constrained number of tests to find ith error after $(i-1)$th detected error
λ_i	4	failure rate of X_i
k_i	5	critical number of consecutive error-free tests for finding ith error
α	5	type I error of test procedure
I	5	stopping time of procedure, *i.e.*, $\min_i\{T_i > k\}$
S_I	8	total number of test runs
q	9	prior distribution of detection probability θ
Θ	10	random detection probability
v	11	$\sum_{i=1}^m (i-1)(t_i-1)$
w	11	$\sum_{i=1}^m (t_i-1)$

Table 1: Table of our notation.

Let X_i $(1 \leq i \leq n)$ be the length of the ith test sequence, *i.e.*, the number of tests to detect the ith error after the $(i-1)$st error has been repaired. Clearly, X_i is geometrically distributed with success parameter $p_i = 1-(1-\theta)^{n-i+1}$. For sake of brevity, we will often write $\varphi = 1-\theta$. We drop the index i when we speak of an arbitrary test run length. Note that the software reliability literature usually deals with continuous-time models. There is a reasonable amount of literature on discrete-time models (see the survey [8]), but discrete failure discounting models like ours seem to be rare.

Recall that the failure rate of a nonnegative integer valued random variable X is defined as

$$\lambda(t) := P(X \leq t \mid X \geq t) = \frac{P(X = t)}{P(X > t-1)}.$$

Thus a geometrically distributed random variable X has failure rate $(1-p)^{t-1}\, p/(1-p)^{t-1} = p$. Hence, the geometric distribution has a constant failure rate, just like its continuous counterpart, the exponential distribution. In our model, we have a stochastic process with a non constant failure rate. After the ith error has been found, the number of test runs until the next detected error is geometrically distributed with success parameter $p_i = 1 - \varphi^{n-i+1}$. Hence, the failure rate in this case equals

$$\lambda_i := \frac{P(X_i = t)}{P(X_i > t-1)} = \frac{\varphi^{(t-1)(n-i+1}\,(1-\varphi^{n-i+1})}{\varphi^{(t-1)(n-i+1)}} = 1 - \varphi^{n-i+1}$$

$$\lambda(s) := \lambda_i \text{ for } \sum_{\ell=1}^{i-1} X_\ell < s \leq \sum_{\ell=1}^{i} X_\ell.$$

This is similar to a discrete-time version of the Jelinski-Moranda model introduced in [9]; a true analogue would have failure rates $\lambda_i = (\lambda(n-i+1))^{-1}$.

In practice unconstrained testing until the next error is detected will not be performed due to budget or time constraints. We therefore assume that there are integers $k_1, k_2, \ldots, k_m$ such that instead of $X_1, X_2, \ldots, X_m$ $(m \leq n+1)$ we observe $T_1, T_2, \ldots, T_m$, where

$$T_i := \begin{cases} X_i & \text{if } X_i \leq k_i \\ k_i + 1 & \text{if } X_i > k_i \end{cases}.$$

When no errors are found in k_i tests, then $T_i := k_i + 1$. So when $T_i = k + 1$, we know that no error has been found in the walk. The probability mass function of a geometric random variable T with success parameter p that is truncated at k is given by

$$P(T = t) = \begin{cases} (1-p)^{t-1}\, p & \text{if } 1 \leq t \leq k \\ (1-p)^{k} & \text{if } t = k+1 \\ 0 & \text{if } t > k+1 \end{cases}. \tag{1}$$

Note that $P(T > \ell) = P(X > \ell)$ for $\ell \leq k$. Hence, we may easily compute the expectation of T:

$$E(T) = \sum_{\ell=0}^{k} P(T > \ell) = \frac{1 - (1-p)^{k+1}}{p}. \tag{2}$$

We now are ready to define the general form of our release procedures. We require a conclusion that all errors have been detected with confidence $1-\alpha$. Typical values for α are 0.05 or 0.10. A straightforward idea would be to estimate n sequentially and test whether n is positive. However, it is known that even for simple models estimators of n are unstable (see *e.g.*, [10]).

Therefore we propose another method. After each observed failure detection, we have a new observed value t_i and test whether the number i of detected errors so far equals the total number of initial errors (n). Because of the one-sided nature of this testing, we test as follows. If $T_i = k_i + 1$, then we conclude that $i = n$, *i.e.*, there are no more errors to be detected. If not, then we continue testing until we find either an error or we do not detect an error during k_{i+1} tests, in which case we declare that our system has no more errors. We now put this in a sequential hypothesis testing framework by using the following hypotheses:

$$H_{0,i}: \; n > i, \quad H_{1,i}: \; n = i.$$

The test statistic is T_i and the critical region is $\{t \mid t > k_i\}$. Note that this way of testing is special, since whenever we do not reject $H_{0,i}$ for some i, we continue testing until we detect another error or we reach the critical bound , *i.e.*, we observe T_{i+1}. Since there are finitely many errors n and we assume perfect repair, our procedure always ends because we stop whenever we do not find an error in k_i tests when $i-1$ errors have already been detected. Hence, the overall number of test runs is at most $n \max\{k_i : i = 1, 2, \ldots, n\}$. Put differently, it is impossible to commit a type II error (not rejecting $H_{0,i}$ when $H_{1,i}$ is true, *i.e.*,, continue testing when there are no more undetected errors). Hence, we only have the possibility of a type I error (rejecting $H_{0,i}$ when $H_{0,i}$ is true, *i.e.*, stop testing when there still are undetected errors). We control the probability of a type I error by choosing the k_i's. Although there is no type II error, we do have an analogue of the notion of power in the form of the total number of tests until we reach a decision.

Alternatively, we could put our procedure in an optimal stopping framework with stopping time I and a special kind of loss function $\mathcal{L}$:

$$I = \min\{i : T_i > k_i\} \text{ and } \mathcal{L} = P_{\theta,n}(I \leq n). \tag{3}$$

Note that the event $\{I = n+1\}$ indicates that all n errors have been detected. Since we are interested in release of error-free systems, our loss function differs from commonly used loss functions (cf. the release procedures mentioned in the introduction).

Using the notation of (3), we may now express the probability of a type I error as

$$\begin{aligned} P_{\theta,n}(I \leq n) = 1 - P_{\theta,n}(I = n+1) &= 1 - P_{\theta,n}(T_1 \leq k_1, \ldots, T_n \leq k_n)\, P_{\theta,n}(T_{n+1} > k_{n+1}) \\ &= 1 - P_{\theta,n}(T_1 \leq k_1, \ldots, T_n \leq k_n), \end{aligned} \tag{4}$$

where the last equality follows from the fact that there are exactly n failures, so that $T_{n+1} = k_{n+1} + 1$ with probability one. Moreover, if k is constant we have

$$P_{\theta,n}(I > \ell) = \prod_{i=1}^{\ell} P_{\theta,n}(T_i \leq k) = \prod_{i=1}^{\ell} (1 - P_{\theta,n}(T_i > k)) = \prod_{i=1}^{\ell} \left(1 - (1-\theta)^{(n-i+1)k}\right). \tag{5}$$

3 Sequential procedure: known detection probability

In this section we assume that θ is known. This is not unrealistic, because we may have seen many similar systems the behaviour of which was observed beforehand. As remarked in

Section 2, we must determine the values of k_i. As a first approximation we take a constant value k for the k_i's. As before, we write $\varphi = 1 - \theta$. Using (5), we then obtain

$$P_{\theta,n}(I \leq n) = 1 - P_{\theta,n}(I > n) = 1 - \prod_{i=1}^{n} \left(1 - \varphi^{(n-i+1)k}\right) = 1 - \prod_{j=1}^{n} \left(1 - \varphi^{jk}\right). \tag{6}$$

In order to have a type I error of at most α, we need to choose k such that $P_{\theta,n}(I \leq n) \leq \alpha$ for all n. Therefore we study the behaviour of $\prod_{j=1}^{n} \left(1 - \varphi^{jk}\right)$. Clearly this product decreases as n increases when both φ and k are fixed, so k must be chosen such that

$$\prod_{j=1}^{\infty} \left(1 - \varphi^{jk}\right) \geq 1 - \alpha. \tag{7}$$

There is no closed expression for the above infinite product. There are several ways to obtain lower bounds. One way is to apply an infinite series expansion known as Euler's Pentagonal Number Theorem (see *e.g.*, [11]):

$$\prod_{j=1}^{\infty} (1 - u^j) = 1 + \sum_{m=1}^{\infty} (-1)^m \left[u^{m(3m-1)/2} + u^{m(3m+1)/2}\right]. \tag{8}$$

Replacing all coefficients (including the zero coefficients) in the sum on the right-hand side of (8) by -1 and adding the trivial upper bound for the left-hand side of (8) by only considering the term with $j = 1$, we obtain the bounds

$$1 - \frac{u}{1-u} = 1 - \sum_{m=1}^{\infty} u^m < \prod_{j=1}^{\infty} (1 - u^j) < 1 - u. \tag{9}$$

This lower bound is negative (and thus useless) for $u > \frac{1}{2}$ and must then be replaced by 0. However, this case cannot occur in the situation considered in this section as we will now show. If we set $u = \varphi^k$ and choose $k = \log_\varphi \alpha$, then the upper bound in (9) shows that $\log_\varphi \alpha$ is a lower bound for the minimal value of k that satisfies (7). Hence, the values of k that we need to consider in (7) are such that $k \geq \log_\varphi \alpha$ which is equivalent to $\varphi^k \leq \alpha$ because $0 < \varphi < 1$. As usual, we only consider values for α that are much smaller than $\frac{1}{2}$. We conclude that the constraint $u < \frac{1}{2}$ is automatically satisfied in our setting.

We obtain tighter, but more complicated bounds, by taking logarithms and using bounds of $\log(1-u)$. For $0 < u < \frac{1}{2}$, this approach yields

$$\exp\left(-\frac{u(u^n - 1)(1 + 2u + u^{n+1})}{u^2 - 1}\right) \leq \prod_{j=1}^{n} (1 - u^j) \leq \exp\left(-\frac{u(u^n - 1)\,(2 + 3u + u^{n+1})}{2(u^2 - 1)}\right) \tag{10}$$

and

$$\exp\left(-\frac{u}{1-u^2}\,(1 + 2u)\right) \leq \prod_{j=1}^{\infty} (1 - u^j) \leq \exp\left(-\frac{u}{1-u^2}\left(1 + \frac{3}{2}u\right)\right). \tag{11}$$

For details, we refer to the appendix. It follows from the proof that the upper bounds are valid for all $0 < u < 1$. Graphical inspection indicate that the lower bounds (10) and (11) hold for $0 < u < 0.684$. We will see later that this suffices for the calculations in this section. Numerical experiments show that for $u < 0.1$, the upper and lower bounds are very close to each other. Tighter lower bounds may be obtained by restricting the range of values: $\log(1-u) > -u - cu^2$ if $0 < u < 1 - \frac{1}{2c}$. However, for our purposes these bounds do not give better results than the bounds in (23).

Note that the left-hand side of (7) is an increasing function of the discrete variable k. Hence, one could determine the minimal value of k such that inequality (7) holds by evaluating the left-hand side of (7) for an increasing sequence of values of k. This evaluation could be done numerically or by using the lower and upper bounds presented in this section. This approach may not appeal to practitioners because one has to perform these numerical searches for each value of α and φ.

There is another method for obtaining values of k such that (7) holds which avoids these numerical searches. This method only requires a small table of corrections to standard values of α, and thus may appeal to practitioners who wish to avoid performing the above calculations. The practitioners only needs the second column of Table 2 and the formula $k = \log_\varphi \alpha$. The rationale behind this approach is as follows. Recall from our discussion after (9) that $\log_\varphi \alpha$ is a lower bound for the minimal value of k that satisfies (7). Note that substitution this value of k into the left-hand side of (7) yields the infinite product $\prod_{j=1}^{\infty} \left(1 - \alpha^j\right)$, which is strictly smaller than $1-\alpha$. Since this infinite product is a strictly decreasing, continuous function of α, there exists for every α such that $0 < \alpha < 1$, a unique number $\tilde{\alpha} < \alpha$ such that $\prod_{j=1}^{\infty} \left(1 - \tilde{\alpha}^j\right) = 1 - \alpha$. Thus $\lceil \log_\varphi \tilde{\alpha} \rceil$ is the minimal value of k such that (7) is satisfied. Table 2 illustrates this procedure for typical type I errors and typical values of φ. Note that $\tilde{\alpha}$ is only slightly smaller than α. Thus the lower bound $\log_\varphi \alpha$ is quite close to the true value k for not too extreme values of φ, as a comparison of Tables 2 and 3 shows. Moreover, computations indicate that $\alpha/(1+\alpha)$, the value that comes from equating the lower bound in (9) with $1-\alpha$, yields a reasonable, practical approximation to $\tilde{\alpha}$.

α	$\widetilde{\alpha}$	φ=0.80	φ=0.85	φ=0.90	φ=0.95	φ=0.99	φ=0.999	φ=0.9999
0.01	0.00985	21	29	44	91	460	4618	46201
0.05	0.04680	14	19	30	60	305	3061	30618
0.10	0.08877	11	15	23	48	241	2421	24216

Table 2: Values of $k = \log_\varphi \tilde{\alpha}$ for fixed detection probability $1-\varphi$.

α	$\varphi = 0.80$	$\varphi = 0.85$	$\varphi = 0.90$	$\varphi = 0.95$	$\varphi = 0.99$	$\varphi = 0.999$	$\varphi = 0.9999$
0.01	21	29	44	90	459	4603	46050
0.05	14	19	29	59	299	2995	29956
0.10	11	15	22	45	230	2302	23025

Table 3: Values of lower bound $k_0 = \log_\varphi \alpha$ for fixed detection probability $1-\varphi$.

Although our procedure always terminates in finite time (*i.e.*, we cannot commit a type II error), it is important to judge its performance by studying $S_I := \sum_{i=1}^{I} T_i$, the total number of tests in our release procedure. Because of our decision rule, we have that our last observation always equals $k+1$. Hence, in order to correctly compute it is essential to use the truncated random variables T_i instead of the X_i's. A martingale argument (for details we refer to the appendix) yields that

$$E_{\theta,n}(S_I) = \sum_{\ell=1}^{n} \left(\frac{1-\varphi^{(k+1)(n-\ell+1)}}{1-\varphi^{n-\ell+1}} \prod_{i=1}^{\ell-1} \left(1-\varphi^{(n-i+1)k}\right) \right) + (k+1) \prod_{j=1}^{n} \left(1-\varphi^{jk}\right). \tag{12}$$

Before we give a table with expected number of tests for several values of α and φ, we first study in detail a small example, since this gives us insight in the above formulas. We take $n = 10$, $\varphi = 0.999$, $\alpha = 0.05$ and choose from Table 2 the corresponding $k = 3061$.

i = number of detected errors after termination of test procedure	expected number of tests given that $I = i+1$	$P(I = i+1)$	contribution to total expected number of tests
0	100	0.000	0
1	212	0.000	0
2	337	0.000	0
3	481	0.000	0
4	648	0.000	0
5	848	0.000	0
6	1099	0.000	0
7	1432	0.000	0
8	1931	0.002	4
9	2885	0.047	135
10	5947	0.951	5656

Table 4: Contributions to expected number of tests for $n = 10$, $\varphi = 0.999$, $k = 3061$ and $\alpha = 0.05$.

Table 4 clearly shows that the main contribution to the expected number of tests comes from the situation when all errors have been detected. This is not surprising, since our test is designed in such a way that with probability 0.95 all errors are detected and we have to wait for k error free tests before we are allowed to terminate the test procedure. There is another striking feature. The largest contribution in case we prematurely terminate our testing procedure comes from the situation when we find all but one errors. Now note that the minimal value of k_0 such that $P_{\theta,1}(T_1 > k_0) = \alpha$ equals $\log_\varphi \alpha$ which, as we have seen before, is a lower bound for the minimal value of k that satisfies (7). This explains why the lower bound $k_0 = \log_\varphi \alpha$ is remarkably accurate (cf. Tables 2 and 3): it is based on the situation that all but the last error have been detected.

We conclude this section with some more tables. The conclusion from Tables 5 and 6 is that the influence of the number of initial errors is relatively small, because a graph of these tables is rather flat. Table 7 shows that it does not make much difference to take a significance level of 5% instead of 10%, while Table 8 shows that expected total testing time is moderate for φ below 0.95 and increases exponentially for larger values of φ.

n	0	10	20	30	40	50	60	70	80	90	100
$E(S_I)$	3062	5795	6468	6871	7159	7385	7570	7728	7866	7988	8098

Table 5: Values of $E(S_I)$ for $\varphi = 0.999$ and $\alpha = 0.05$.

n	200	300	400	500	600	700	800	900	1000
$E(S_I)$	8839	9296	9636	9912	10149	10358	10548	10723	10886

Table 6: Values of $E(S_I)$ for $\varphi = 0.999$ and $\alpha = 0.05$.

α	0.01	0.05	0.10
$E(S_I)$	7496	5795	5020

Table 7: Values of $E(S_I)$ for $\varphi = 0.999$ and $n = 10$.

φ	0.80	0.85	0.90	0.95	0.99	0.999	0.9999
$E(S_I)$	34	43	63	120	583	5795	57911

Table 8: Values of $E(S_I)$ for $n = 10$ and $\alpha = 0.05$.

4 Sequential procedure: unknown detection probability

In this section we assume that θ is unknown. We model this situation by assuming that $\varphi = 1 - \theta$ is a continuous random variable on $[0, 1]$ with density q. Unfortunately, we have been unable to compute the type I error of a decision procedure where the prior distribution of φ is updated after each detected failure. Therefore we restrict ourselves in this section to computations of our procedure without update. A Bayesian one-stage testing procedure (*i.e.*, we update only once) is studied in Section 5. The one-stage testing procedure applied as a fully adaptive sequential procedure (*i.e.*, we repeatedly update) is studied through extensive simulations in Section 6.

As remarked in Section 2, we must determine the values of k_i. As in the previous section, we take a constant value k for the k_i's. Using (5) and (6), we then obtain

$$P_n(I \leq n) = \int_0^1 P_{\varphi,n}(I \leq n)\, q(\varphi)\, d\varphi = \int_0^1 \left(1 - \prod_{j=1}^{n} \left(1 - \varphi^{jk}\right)\right) q(\varphi)\, d\varphi. \tag{13}$$

In order to have a type I error of at most α, we need to choose k such that $P_n(I \leq n) \leq \alpha$ for all n. Although the right-hand side of (13) is monotone in n, we cannot apply the sharp bounds (11) because the support of q typically contains values close to 1. A pragmatic approach is to fix a reasonably large value of n and compute exact values of k corresponding to this value of n. Experimentation shows that computing the type I error for these exact values of k for different values of n quickly leads to accurate manual determination of values for k for all n.

A suitable class of prior distributions for the passage probability φ is the class of uniform distributions on an interval near 1. Another suitable class of prior distributions is the Beta distribution. Beta distributions have a density $t^{\gamma-1}(1 - t^{\delta-1})/B(\gamma, \delta)$ for $0 \leq t \leq 1$, where

$B(\gamma, \delta)$ is a normalizing constant. In order to avoid having too much mass near 1, we choose Beta distributions the density of which has slope zero at $t = 1$. For sake of convenience, we choose $\delta > 2$. Since the mean equals $\gamma/(\gamma + \delta)$, we may obtain an appropriate mean by suitably choosing γ. We conclude from Table 9 that the values of k are considerably larger than those in the tables of the previous section. The influence of letting the support of φ to be too close to 1 is considerable. The values of k for Beta distributions are much smaller than those of uniform distributions with the same mean, because the Beta distributions have smaller variance.

distribution	mean	$\alpha = 0.01$	$\alpha = 0.025$	$\alpha = 0.05$	$\alpha = 0.10$
Uniform(0.90,0.95)	0.9250	66	52	42	33
Uniform(0.90,0.98)	0.9400	118	89	69	51
Uniform(0.90,0.99)	0.9450	185	132	98	68
Uniform(0.90,0.999)	0.9495	653	362	211	115
Uniform(0.95,0.99)	0.9700	238	179	138	103
Uniform(0.95,0.999)	0.9745	944	576	364	212
Uniform(0.80,1)	0.9000	627	251	125	62
Uniform(0.85,1)	0.8750	836	334	167	83
Uniform(0.90,1)	0.9500	1255	502	251	125
Uniform(0.95,1)	0.9700	2510	1004	502	251
Uniform(0.96,1)	0.9800	3138	1255	627	313
Uniform(0.98,1)	0.9900	6276	2510	1255	627
Uniform(0.99,1)	0.9950	12551	5020	2510	1255
Uniform(0.999,1)	0.9995	125519	50207	25103	12551
Beta(27,3)	0.9000	109	73	53	37
Beta(57,3)	0.9500	225	152	110	76
Beta(147,3)	0.9800	573	386	279	194
Beta(297,3)	0.9900	1154	778	561	389
Beta(20,1.05)	0.9500	1979	816	413	204
Beta(20,1.1)	0.9480	1604	687	357	182
Beta(30,1.05)	0.9662	2968	1224	619	306
Beta(30,1.1)	0.9646	2404	1029	535	272

Table 9: Values of k for various prior distributions.

5 Bayesian one-stage testing

We study in this section a Bayesian one-stage testing version of our release procedure. We have the following application in mind. Suppose that a company has performed testing on software and kept records. In order to obtain an official certificate that the software has no errors, the software must be tested by an official agency. The computations of this section are also useful as an intermediate step for the adaptive multi-stage testing procedure in Section 6.

As said before we apply here a mixture of Bayesian and classical statistics. We do not assume a prior distribution for the total number of initial errors n, since we think it is unrealistic to assume that a good elicitation procedure for n exists in practice. Instead, we use a worst-case scenario: the maximal type I error as a function of n should be small. However, $\varphi = 1-\theta$ (the passage probability of errors) is assumed to be a random variable with density q. In this way we are able to learn from previous system testing and also from the testing of the system under consideration. We assume that we know the test history, *i.e.*, we know both m, the number of detected errors that have been detected and the number of tests already executed to detect each of the m detected errors.

In order to have a type I error not exceeding α, we must find a critical value k such that

$$\begin{aligned}\alpha &\geq \max_{n\geq m+1} P_{\varphi,n}(X > k \mid X_1 = t_1, \ldots, X_m = t_m) \\ &= \max_{n\geq m+1} \frac{\int_0^1 P_{\varphi,n}(X > k, X_1 = t_1 \ldots, X_m = t_m)\, q(\varphi) d\varphi}{\int_0^1 P_{\varphi,n}(X_1 = t_1 \ldots, X_m = t_m)\, q(\varphi) d\varphi}.\end{aligned} \tag{14}$$

Note that the above formula is expressed in terms of X_i's instead of T_i's, because these observations have not been subjected to the stopping criterion I. Clearly the critical value k is a function of $t_1, \ldots, t_m$ only. Note that $X_1, X_2, \ldots, X_m, X$ are independent, given φ. Hence, the above inequality can be rewritten as

$$\begin{aligned}\alpha \quad &\geq \quad \max_{n\geq m+1} \frac{\int_0^1 \varphi^{(n-m)k} \prod_{i=1}^m \varphi^{(n-i+1)(t_i-1)}(1-\varphi^{n-i+1})\, q(\varphi)\, d\varphi}{\int_0^1 \prod_{i=1}^m \varphi^{(n-i+1)(t_i-1)}(1-\varphi^{n-i+1})\, q(\varphi)\, d\varphi} \\ &= \quad \max_{n\geq m+1} \frac{\int_0^1 \varphi^{(n-m)k+nw-v} \prod_{i=1}^m (1-\varphi^{n-i+1})\, q(\varphi)\, d\varphi}{\int_0^1 \varphi^{nw-v} \prod_{i=1}^m (1-\varphi^{n-i+1})\, q(\varphi)\, d\varphi}\ ,\end{aligned} \tag{15}$$

where

$$w_i = t_i - 1,\ w = \sum_{i=1}^m w_i,\ v = \sum_{i=1}^m (i-1) w_i. \tag{16}$$

In order to obtain computable lower bounds for k, we apply the following general form of Chebyshev's inequality.

Lemma 5.1 (Chebyshev's Inequality) *If f is an integrable non-increasing function, g is an integrable non-decreasing function and p is an integrable function, then*

$$\frac{\int_0^1 p(x)f(x)g(x)\, dx}{\int_0^1 p(x)f(x)\, dx} \leq \frac{\int_0^1 p(x)g(x)\, dx}{\int_0^1 p(x)\, dx}. \tag{17}$$

If both f and g are integrable non-decreasing or non-increasing functions, and p is an integrable function, then

$$\frac{\int_0^1 p(x)f(x)g(x)\, dx}{\int_0^1 p(x)g(x)\, dx} \geq \frac{\int_0^1 p(x)f(x)\, dx}{\int_0^1 p(x)\, dx}. \tag{18}$$

For a proof and a detailed discussion of this lemma, we refer to [12, Chapter 9] .

If Y is a continuous random variable on $[0,1]$ and i, j are nonnegative integers, then we may bound the moments as follows. The first inequality is a direct corollary of (18), while the second one is trivial because Y is defined on $[0,1]$:

$$E(Y^i)\, E(Y^j) \leq E(Y^{i+j}) \leq E(Y^j). \tag{19}$$

Applying (17) to (15) with $f(\varphi) = \varphi^{(n-m)k}$, $g(\varphi) = \prod_{i=1}^{m}(1-\varphi^{n-i+1})$, and $p(\varphi) = q(\varphi)\,\varphi^{nw-v}$, we obtain

$$\begin{aligned} \max_{n\geq m+1} \frac{\int_0^1 \varphi^{(n-m)k+nw-v}\prod_{i=1}^{m}(1-\varphi^{n-i+1})\,q(\varphi)\,d\varphi}{\int_0^1 \varphi^{nw-v}\prod_{i=1}^{m}(1-\varphi^{n-i+1})\,q(\varphi)\,d\varphi} &\leq \max_{n\geq m+1} \frac{\int_0^1 \varphi^{(n-m)k+nw-v}\,q(\varphi)\,d\varphi}{\int_0^1 \varphi^{nw-v}\,q(\varphi)\,d\varphi} \\ &= \max_{n\geq m+1} \frac{EY^{(n-m)k+nw-v}}{EY^{nw-v}} \\ &= \max_{n\geq m+1} \frac{EY^{n(w+k)-(mk+v)}}{EY^{nw-v}}, \qquad (20) \end{aligned}$$

where Y denotes a random variable with density q. Denote the right-hand side of (20) by $L(n,k)$. The function $L(n,k)$ is not always increasing in n for fixed k (*e.g.*, consider a two-point distribution on $[0,1]$ with positive mass in 1). However, for Beta distributions (including the uniform distribution, which is a special case), the function $L(n,k)$ has the desired monotonicity property as we will show below. We have not been able to prove this property in general for uniform distribution on $[\ell,1]$ for all $0<\ell<1$. Calculations seems to indicate that the property holds for all values of ℓ, except those that are very close to 1 (depending on the values of m, k, w and v).

Computation of k in case of a Beta(γ,δ) distribution

A straightforward calculation (see Appendix) shows that (20) is maximal for $n = m+1$. Hence, we need to choose k such that

$$\alpha \geq \frac{EY^{k+(m+1)w-v}}{EY^{(m+1)w-v}} = \frac{\gamma^{[k+(m+1)w-v]}}{\gamma^{[(m+1)w-v]}}\,\frac{(\gamma+\delta)^{[(m+1)w-v]}}{(\gamma+\delta)^{[k+(m+1)w-v]}}, \qquad (21)$$

where $\gamma^{[j]} = \gamma(\gamma+1)\ldots(\gamma+j-1)$ denotes the Pochhammer symbol. In case of a uniform distribution on $[0,1]$ (*i.e.*, $\gamma=\delta=1$), the above condition simplifies to

$$\alpha \geq \frac{EY^{k+(m+1)w-v}}{EY^{(m+1)w-v}} = \frac{(m+1)w-v+1}{k+(m+1)w-v+1}\,\frac{1-\ell^{k+(m+1)w-v+1}}{1-\ell^{(m+1)w-v+1}}. \qquad (22)$$

Results on the performance of the Chebyshev bounds can be found in the next section, where we repeatedly apply the one-stage Bayesian testing to obtain an adaptive procedure.

6 Adaptive testing

In this section we extend the procedures of the previous section to a fully adaptive testing procedure. We assume a prior distribution on the detection probability. The value of k, the threshold for the nummer of error-free tests, is updated after each detected error as if we were performing one-stage testing. An analytical derivation of optimal values of k that guarantee an overall type I error seems to be untractable. Therefore we performed extensive simulations to check the performance of this procedure. Since we repeatedly apply k values that correspond to one-stage testing, we expect an increase of the type I error. We hope that this inflation is limited, so that our procedure could be used in practice without serious problems.

We checked the performance of two update rules: an approximation rule assuming that the maximum in (14) is attained at $n = m + 1$ and a rule based on the Chebyshev rule (20), which can be shown to be maximal at $n = m + 1$ for the uniform and Beta distributions that we consider. As a benchmark, we also consider a rule with fixed k as in Section 4.

The simulation was executed in a layered way. As prior distributions we took the uniform distributions on $[0.8; 1]$, $[0.9; 1]$ and $[0.96; 1]$ as well as the Beta distributions with parameters 27 and 3. As initial number of errors we took 2, 5 and 10. Instead of directly generating detection probabilities from the prior distributions, we discretized the support of the distribution into 40 points. For each value of the detection probability in the discretized support, we generated 5000 test histories. The type I error is computed by a numerical integration of the discretized type I errors. Experimentation showed that a grid of 40 points is sufficient to obtain accurate results. The layered approach allowed us to control the effect of small detection probabilities during the simulation. We choose to replicate 5000 times, because for smaller values the standard deviation of the estimated proportion of test histories in which all errors are detected is not small enough compared to a type I error of 0.05. In order to compensate for the expected inflation of the type I error, we targeted our procedures at a type I error of $\alpha/(1 + \alpha)$ instead of α in order to have a simple minor compensation. Of course, other ways to compensate are possible too.

We recorded not only the observed type I errors during the simulation, but also summary statistics on the number of not detected errors, the decision thresholds k_i as well as the total number of tests.

The first conclusion is that the Chebyshev method is much too conservative. As a consequence, it requires an extremely high number of test runs. Hence, this method is not feasible in practice. The increase of the type I error for the approximation rule was reasonable for the uniform distributions and quite good for the Beta distribution. The inflation generally increases with increasing type I error. The observed type I error for $n = 5$ was generally somewhat higher than for $n = 2$ and $n = 10$.

	$n = 2$	$n = 5$	$n = 10$
$\alpha = 0.01$	0.014	0.016	0.012
$\alpha = 0.025$	0.036	0.042	0.034
$\alpha = 0.05$	0.070	0.083	0.071

Table 10: Uniform$(0.8, 1)$ observed α for corrected α.

	$n = 2$	$n = 5$	$n = 10$
$\alpha = 0.01$	0.014	0.016	0.013
$\alpha = 0.025$	0.036	0.042	0.034
$\alpha = 0.05$	0.070	0.084	0.072

Table 11: Uniform$(0.9, 1)$ observed α for corrected α.

Proof of (10) and (11): Truncated Taylor series for $\log(1 - x)$ yield upper bounds for

	$n = 2$	$n = 5$	$n = 10$
$\alpha = 0.01$	0.014	0.016	0.013
$\alpha = 0.025$	0.036	0.042	0.035
$\alpha = 0.05$	0.070	0.085	0.073

Table 12: Uniform$(0.96, 1)$ observed α for corrected α.

	$n = 2$	$n = 5$	$n = 10$
$\alpha = 0.01$	0.011	0.012	0.011
$\alpha = 0.025$	0.028	0.032	0.028
$\alpha = 0.05$	0.0547	0.064	0.058

Table 13: Beta(27,3) observed α for corrected α.

$\log(1 - x)$ that holds for $0 < x < 1$. Lower bounds may be verified easily considering derivatives. *E.g.*, for $0 < x < \frac{1}{2}$ we have:

$$-x - x^2 < \log(1 - x) \leq -x - \frac{x^2}{2}. \tag{23}$$

It follows from applying (23) that

$$\log \prod_{j=1}^{n} \left(1 - \alpha^j\right) = \sum_{j=1}^{n} \log\left(1 - \alpha^j\right) \leq \sum_{j=1}^{n} \left(-\alpha^j - \frac{\alpha^{2j}}{2}\right) = -\frac{\alpha(1 - \alpha^n)(2 + 3\alpha + \alpha^{n+1})}{2(1 - \alpha^2)}.$$

In a similar way we obtain a lower bound. By letting n go to infinity we obtain the bounds for the infinite product. □

Proof of (12) Write $M_k = \sum_{i=1}^{k} (T_i - E(T_i))$. Note that the sequence $M_1, \ldots, M_n$ is a martingale with respect to $T_1, \ldots, T_n$, *i.e.*, $E_{\theta,n}(|M_\ell|) < \infty$ and $E_{\theta,n}(M_{\ell+1}|T_1, T_2, \ldots, T_\ell) = M_\ell$. Obviously I is a stopping time, *i.e.*, the events $I = \ell$ is in the σ-algebra generated by the random variables $T_1, \ldots, T_\ell$. Thus by the Optional Stopping Theorem (see *e.g.*, [13, Sec. 7.9] we have $E_{\theta,n}(M_I) = E_{\theta,n}(M_1) = 0$. Thus, writing out the definition of M_k we find that

$$E_{\theta,n}(S_I) = E\left(\sum_{\ell=1}^{I} T_\ell\right) = E\left(\sum_{\ell=1}^{I} E(T_\ell)\right). \tag{24}$$

Note that the Optional Stopping Theorem has simplified our task, because the randomness in the right-hand side of (24) is reflected through I only. Hence, writing $p_i := 1 - (1 - \theta)^{n-i+1} =$

$1-\varphi^{n-i+1}$ and using that $T_{n+1}=k+1$, we obtain

$$\begin{array}{lr}
E_{\theta,n}(S_I) & = \\
E\left(\sum_{\ell=1}^{I} E(T_\ell)\right) = \sum_{i=1}^{n+1}\left(\sum_{\ell=1}^{i} E(T_\ell)\right) P(I=i) & = \\
\sum_{i=1}^{n}\left(\left\{\sum_{\ell=1}^{i} \frac{1-(1-p_\ell)^{k+1}}{p_\ell}\right\} P(I=i)\right) + \left\{k+1+\sum_{\ell=1}^{n} \frac{1-(1-p_\ell)^{k+1}}{p_\ell}\right\} P(I=n+1) & = \\
\sum_{\ell=1}^{n}\left(\frac{1-(1-p_\ell)^{k+1}}{p_\ell} \sum_{i=\ell}^{n} P(I=i)\right) + \left\{k+1+\sum_{\ell=1}^{n} \frac{1-(1-p_\ell)^{k+1}}{p_\ell}\right\} P(I=n+1) & = \\
\sum_{\ell=1}^{n}\left(\frac{1-(1-p_\ell)^{k+1}}{p_\ell} P(I>\ell-1)\right) + (k+1)\, P(I=n+1) & = \\
\sum_{\ell=1}^{n}\left(\frac{1-\varphi^{(k+1)(n-\ell+1)}}{1-\varphi^{n-\ell+1}} \prod_{i=1}^{\ell-1} \left(1-\varphi^{(n-i+1)k}\right)\right) + (k+1) \prod_{j=1}^{n} \left(1-\varphi^{jk}\right). &
\end{array}$$

□

Remark .1 *There are more useful martingales:* $S_n^2 - n\,\sigma^2$ *may be used to compute* $Var(S_I)$*. The exponential martingale* $e^{tS_n} / \left(\prod_{i=1}^{n} E(e^{t\,T_i})\right)$ *may be used to obtain expressions for the moment generating function of* S_I*.*

Proof of (21) We will use that $\Gamma(x+j) = x^{[j]}\Gamma(x)$ for $j \in \mathbb{N}$ and that the Beta function equals $B(\gamma,\delta) = \frac{\Gamma(\gamma)\Gamma(\delta)}{\Gamma(\gamma+\delta)}$. The jth moment of the Beta (γ,δ) distribution equals

$$\frac{B(\gamma+j,\delta)}{B(\gamma,\delta)} = \frac{\Gamma(\gamma+j)}{\Gamma(\gamma)}\frac{\Gamma(\gamma+\delta)}{\Gamma(\gamma+j+\delta)} = \frac{\gamma^{[j]}}{(\gamma+\delta)^{[j]}},$$

where $\gamma^{[j]} = \gamma(\gamma+1)\ldots(\gamma+j-1)$ denotes the Pochhammer symbol. Hence, (20) becomes

$$\begin{aligned}
\frac{B(\gamma+n(w+k)-(mk+v),\delta)}{B(\gamma+\delta)}\frac{B(\gamma+\delta)}{B(\gamma+nw-v,\delta)} &= \frac{\gamma^{[n(w+k)-(mk+v)]}}{(\gamma+\delta)^{[n(w+k)-(mk+v)]}}\frac{(\gamma+\delta)^{[nw-v]}}{\gamma^{[nw-v]}} \\
&= \frac{\gamma^{[nw-v+k(n-m))]}}{\gamma^{[nw-v]}}\frac{(\gamma+\delta)^{[nw-v]}}{(\gamma+\delta)^{[nw-v+k(n-m))]}}.
\end{aligned}$$

Cancelling terms and changing the order of the remaining terms, we obtain the following expression

$$\frac{(\gamma+nw)(\gamma+nw+1)\ldots(\gamma+nw-v+k(n-m)-1)}{(\gamma+\delta+nw)(\gamma+\delta+nw+1)\ldots(\gamma+\delta+nw-v+k(n-m)-1)}$$

The above expression is a product of terms of the form

$$\frac{\gamma+nw+i}{\gamma+\delta+nw+i} = 1 - \frac{\delta}{\gamma+\delta+nw+i},$$

which is a decreasing function of n since $\gamma>0$, $\delta>0$ and $w>0$. □

References

[1] C. Lee, K. Nam, and D. H. Park, "Optimal software release policy based on Markovian perfect debugging model," *Comm. Statist. Theory Methods*, vol. 30, no. 11, pp. 2329–2342, 2001, International Conference on Statistics in the 21st Century (Orono, ME, 2000).

[2] N. Morali and R. Soyer, “Optimal stopping in software testing,” *Naval Res. Logist.*, vol. 50, no. 1, pp. 88–104, 2003.

[3] S. Özekici, İ. K. Altínel, and E. Angün, “A general software testing model involving operational profiles,” *Probab. Engrg. Inform. Sci.*, vol. 15, no. 4, pp. 519–533, 2001.

[4] S. Özekici, İ. K. Altínel, and S. Özçelikyürek, “Testing of software with an operational profile,” *Naval Res. Logist.*, vol. 47, no. 8, pp. 620–634, 2000.

[5] S. Özekici and R. Soyer, “Bayesian testing strategies for software with an operational profile,” *Naval Res. Logist.*, vol. 48, no. 8, pp. 747–763, 2001.

[6] G. Becker and L. Camarinopoulos, “A Bayesian estimation method for the failure rate of a possibly correct program,” *IEEE Trans. Soft. Eng.*, vol. 16, no. 11, pp. 1307–1310, 1990.

[7] T. Ferguson and J. Hardwick, “Stopping rules for proofreading,” *J. Appl. Prob.*, vol. 26, pp. 304–313, 1989.

[8] A. Fries and A. Sen, “A survey of discrete reliability-growth models,” *IEEE Trans. Rel.*, vol. 45, pp. 582–604, 1996.

[9] Z. Jelinski and P. Moranda, “Software reliability research,” in *Statistical Computer Performance Evaluation*, W. Freiberger, Ed. Academic Press, 1972, pp. 465–497.

[10] H. Joe and N. Reid, “Estimating the number of faults in a system,” *J. Amer. Stat. Assoc.*, vol. 80, no. 389, pp. 222–226, 1985.

[11] C. Berge, *Principles of combinatorics*, ser. Translated from the French. Mathematics in Science and Engineering, Vol. 72. New York: Academic Press, 1971.

[12] D. Mitrinović, J. Pečarić, and A. Fink, *Classical and new inequalities in analysis*, ser. Mathematics and its Applications (East European Series). Dordrecht: Kluwer Academic Publishers Group, 1993, vol. 61.

[13] G. Grimmett and D. Stirzaker, *Probability and random processes*, 2nd ed. New York: The Clarendon Press Oxford University Press, 1992.