

Estimating Bounds on the Reliability of Diverse Systems

Peter Popov[†], Lorenzo Strigini[†], John May[‡], Silke Kuball[‡]

[†]Centre for Software Reliability, City University
Northampton Square, London, EC1V 0HB, UK

[‡]Safety Systems Research Centre, University of Bristol
Woodland Road, Bristol, BS8 1UB, UK

(This version corresponds to Version 1.0, 31 May 2000, of the DISPO Technical Report of the same title, PP-DD-TR-03-v1_0)

Abstract

We address the problems of estimating the reliability of multiple-version software, and improve the understanding of the various ways failure dependence between versions can arise. Specifically, we step from the previous conceptual models, which described what behaviour could be expected "on average" from a randomly chosen pair of "independently generated" versions to predictions using specific information about a given pair of versions. The concept of "variation of difficulty" between situations to which software may be subject is central to the previous models cited. We show that it has more far-reaching implications than previously found.

We show the practical implications of varying probabilities of failure over input subdomains or operating regimes. A direct practical gain for designers, users and regulators is the possibility of estimating useful upper and lower bounds on the reliability of a two-versions system.

Key words: Software fault-tolerance, design diversity, demand space partitioning, common-mode failure.

1. Introduction

Design diversity is an intuitively attractive method for increasing the reliability of critical systems subject to design error. In applications such as nuclear plant protection it is common to employ parallel, diverse "channels" that separately process plant data and trigger a safe shut-down if they detect indications of unsafe conditions. With control or safety systems that use software or complex computer hardware (like modern microprocessors), where design errors are widespread, design diversity appears even more desirable and has achieved some degree of recognition from industry and some regulators [1], [2]. Additional interest has arisen recently due to two trends. The first is that for some applications that traditionally relied on non-software-based systems, at least as backups for software-based ones, this alternative has disappeared: the necessary hardware components are no longer available. The second trend is the increased reliance on commercial-off-the-shelf (COTS) systems or subsystems. These are often developed without the rigorous quality assurance procedures required by customers with high reliability requirements. Then, it seems desirable to add to the COTS product a backup system, which would detect and contain especially dangerous failures. However, the use of diversity remains controversial because of the difficulty of evaluating its advantages (and, as a corollary, of managing its application so as to achieve dependability in a cost-effective manner).¹

¹ We note in passing that in this respect software design diversity has been researched more thoroughly than most software engineering techniques. Ironically, this very effort, which has accumulated evidence that diversity is useful [1], [7], [10], has led many (through misunderstanding of the Knight and Leveson results [10]) to believe that it is not.

Design diversity, in the basic form used in industry, requires that two or more versions of a program be developed by separate teams, which do not communicate, and may be given explicit directives for diversifying the internal structures of their products (e.g., using different algorithms), while keeping the products equivalent in their externally visible functional behaviour. It is hoped that, if one version fails, the other, being internally different, will not fail at the same time: if they contain bugs, these will not cause failures in exactly the same circumstances in all versions. The two or more versions are then run in a redundant configuration, so that failures in a subset of the versions may be masked or at least detected. More refined arrangements are possible, e.g. with some version performing a monitoring or auditing function but no control function.

Estimating the dependability of diverse-redundant systems before they are used is crucial, as they are usually safety-critical systems. In addition, their requirements are usually in the class of "ultra-high reliability" [3], for which black-box operational testing cannot produce sufficiently high confidence at a feasible cost.

The difficulty in reliability modelling of diverse-redundant systems hinges on the fact that one cannot claim *independence of failures* in diverse versions. Independence would allow claims like: "The probability of failure per demand of a 2-channel safety system, in which each channel has been convincingly shown to have a probability of failure per demand not greater than 10^{-3} , is not greater than $(10^{-3})^2 = 10^{-6}$ ". Without independence, this is not possible. Independence is often postulated between identical hardware channels, concerning failures due to physical (non-design) faults. For design faults, we have convincing evidence that independence should not be expected to arise naturally. Conceptual models, first developed by Eckhardt and Lee [4], support this view. A similar result actually applies to physical faults. Parts of the reliability engineering community routinely account for it by rules-of-thumb which limit the reliability increase that one is allowed to claim through the use of redundancy. A theoretical explanation of positive correlation among physical failures is due to Hughes [5]. Littlewood [6] summarizes and compares the models concerning design faults and physical faults. We will use the terminology of software and design faults, though many considerations apply to hardware and physical faults as well.

In this paper, we extend previous conceptual models to improve our understanding of failure behaviour. We gain a small improvement in the ability actually to predict reliability, and a greater one in our ability to avoid misleading predictions. Previous conceptual models dealt mostly with what could be expected "on average" from the failure behaviour of diverse versions developed for a given application. We discuss the problem of prediction "in particular", for an *individual set of versions*, i.e., the prediction that is truly important for a designer, customer or regulator. In detail, we study the probability of failure on demand for a 2-version system.

In section 2 we recall the essential previous results in modelling diverse-redundant systems. Section 3 describes the reliability of a two-version system in terms of complete knowledge of which demands are failure demands for either version. Section 4 shows how to apply this model on the basis of incomplete knowledge, which may be available in practice, i.e. knowledge about failures of the versions over *subdomains* of the demand space. In section 5 we illustrate the derivation of lower and upper bounds for the probability of system failure using data from an experiment on multi-version software. Section 6 discusses the practical implications of these results.

2. Background

The problem of evaluating the reliability of fault-tolerant software is far from being solved, despite the long time since the idea of improving the reliability of software through redundancy was introduced. The initial hope of building up a software system with arbitrarily high reliability from components with modest reliability has been proven wrong by a number

of empirical studies, summarised recently in [7], in which failures of "independently developed" versions were positively correlated. More importantly, the break-through modelling work by Eckhardt and Lee [4] ("EL model") showed that these empirical results were not artefacts but very much what one should expect. These authors assumed "independently developed" versions in the sense that, given a *specific* demand, the failure of one version would not give any indication about the probability of failure of another version, randomly chosen among the infinite other possible versions.

The problem solved by different software versions is common to all of them. Different parts of the problem present different degrees of difficulty, and this phenomenon makes independent teams more likely to err while building the same "difficult" parts of the problem than they are on average. Therefore multiple "independent" developments may fail to produce independent failures between versions (when failure probability is averaged over *all* demands). For a randomly chosen pair of versions (i.e. "on average" over all possible pairs) there will be positive correlation for most plausible form of "problem difficulty". Littlewood and Miller [8] later showed ("LM model") that with forced design diversity the varying difficulty could be turned from a disadvantage into a benefit. "Forced" diversity means imposing different constraints on the different teams, e.g. different design methods, tools and languages. If what is difficult for one team is consistently easy for another team, the actual reliability of a pair of versions "on average" will be better than what one can expect under the assumption of independence.

Both these models are "conceptual" models only, in that they do not support predictions for specific cases, and very much depend on the notion of "difficulty" defined over the space of the demands (input domain) of the software². The difficulty of a demand point x is quantified by the probability $\theta(x)$ that a version, randomly chosen among those that at least hypothetically can be developed from the same specification, will fail on x . If the demand is "intrinsically difficult" then $\theta(x)$ will be close to 1. If demand x_1 is more difficult than x_2 then $\theta(x_1) > \theta(x_2)$. Presumably, for the vast majority of demands $\theta(x)$ will be close to 0.

Under the EL model the variation of difficulty over the demand space is deemed an inevitable disadvantage, while under the LM model it may turn out to be an advantage. Everything depends on the covariance between the random variables Θ_A and Θ_B representing the difficulty under two diverse design processes A and B, respectively. Θ_A is by definition a function $\theta_A(X)$ representing the difficulty of a randomly selected demand X when design process A is used. The work by Littlewood and Miller [8] indicates the need for research to objectively measure development processes in quest for less (in the best case negatively) correlated processes. To the best of our knowledge, little progress has been reported in this direction so far. Nicola and Goyal [9] suggested the Beta distribution as a suitable representation of the probability of failure of a randomly selected version on a randomly selected demand, with the two model parameters to be determined from testing. These authors refer to the experiment by Knight and Leveson [10] to show the plausibility of the suggested distribution.

The problem with using these two conceptual models in practice is two-fold. On the one hand, one can hardly expect the "difficulty" function Θ to be measurable. On the other hand, even if it were, the predictions of the models are predictions for an "average" multiple-version development. The models consider all software realisations from the same specification. If we could repeat many times the process of developing pairs of versions and take the average over their probabilities of coincident failure, we would see the numbers predicted by the models. The models say nothing about a *particular* pair of versions,

² An "input", in this context, is one demand on the software. For instance, for a plant protection system it could be the whole sequence of sensor inputs as the plant state evolves to cross its safe operation envelope.

however. A specific software project will only develop a small number of versions (typically 2, 3 or 4). What is of interest is the reliability of the *particular* set of developed versions in hand. The EL and LM models tell us that the independence assumption about coincident failure of versions may be wrong. We can not even say whether it is optimistic or pessimistic. The experiment by Knight and Leveson [10] confirms that individual pairs may greatly vary in their probability of coincident failure and deviate from the result "on average". Similar results were observed in other empirical studies recently summarized in [7]. In a simulation study this phenomenon was exercised [11] and the results, under plausible assumptions, were very discouraging: with increasing reliability, the result "on average" becomes a poorer and poorer indication of the reliability "in particular".

In addition to the EL and LM models, a series of "structural" models for fault-tolerant software have been published ([12], [13], [14], [15], [16], [17]) which do not attempt to predict the likelihood of common failure on one demand, but to predict the reliability of a diverse-redundant system over time, given parameters that describe the behaviours of its components, e.g., the probabilities of individual and common failures per execution step, the duration of execution steps, etc. These models are straightforward, though complex, applications of Markov chains, and allow a user to understand the respective roles of the various parameters. However, they are useless for actual prediction, because the many parameters they require, especially the probabilities of common failures of versions, are difficult to estimate.

We attempt to bridge the gap between these two kinds of models, by studying how the conceptual model of failure generation can be applied to a specific set of versions.

For the sake of simplicity, all our models refer to the simplest possible diverse-redundant configuration: two versions, with perfect adjudication ("1-out-of-2", diverse system). Although very simple, this scheme corresponds to some real systems, like a plant safety shut-down system, in which two versions run on completely separate and non-communicating hardware channels (sensors, computers and actuators), and either version is able to order a shut-down action no matter what the other does.

3. Probability of failure on demand in 2-version system, given complete knowledge

A 1-out-of-2 system is said to fail if both versions produce incorrect results on the same demand. It is said to succeed otherwise. We call the two versions A and B.

For each version, we define a score (indicator) function. $\omega_A(x)$ and $\omega_B(x)$ represent the scores of version A and B on demand x , respectively. Saying that, for instance, $\omega_A(x) = 1$ means that version A deterministically fails on demand x . We can define two random variables, Ω_A and Ω_B , describing the behaviours of A and B on a random demand X , as:

$$\Omega_A = \omega_A(X), \quad \Omega_B = \omega_B(X) \quad (1)$$

Then the probabilities of failure of versions A and B on a randomly selected demand X (probability of failure on demand, *pdf*) are:

$$P_A \equiv P(A \text{ fails on } X) = E(\Omega_A) = \sum_{x \in D} P(x) \cdot \omega_A(x)$$

and

$$P_B \equiv P(B \text{ fails on } X) = E(\Omega_B) = \sum_{x \in D} P(x) \cdot \omega_B(x) \quad (2)$$

where D denotes the demand space and $P(x)$ is the probability that demand x will be input to the software (the demand profile of the software).

By definition:

$$\text{cov}(\Omega_A, \Omega_B) = \sum_{x \in D} (\omega_A(x) - P_A) \cdot (\omega_B(x) - P_B) \cdot P(x) \quad (3)$$

The probability of coincident failure on one randomly chosen demand X can then be expressed as:

$$P(A, B \text{ fail on } X) = P(X \text{ such that } \omega_A(X) = \omega_B(X) = 1) =$$

$$P(A \text{ fails on } X)P(B \text{ fails on } X) + \text{cov}(\Omega_A, \Omega_B) \quad (4)$$

The expression given by (4) appears very similar to the expression for the probability of system failure given by the LM model. The difference, however, is huge. In the LM model the difficulty (Θ_A and Θ_B) as a random variable is expressed by the probability that a version (picked at random from the population of possible versions produced from the same specification using the same design process) will fail on a randomly selected demand. The values that this difficulty can take on particular demands may be any number in the range $[0, 1]$ including the extreme values. In (4), Ω_A and Ω_B represent the *score functions* of two particular versions. The values that these two random variables can take on randomly selected demands are strictly from the set $\{0, 1\}$, and no value in between. The terms $P(A \text{ fails on } X)$ and $P(B \text{ fails on } X)$ in (4) truly represent the probabilities of failure of the two particular versions, while the similar terms under the LM model represent the probabilities of two *randomly selected* versions developed using different design processes. This is why we say that the LM probabilities of failure are measures “on average”.

We can express the distribution of Ω_A (or Ω_B) as something like Fig. 1a:

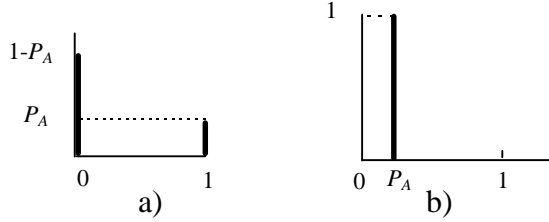


Fig. 1

The X-axis represents the values of the score function (probability of failure on one demand). On the Y-axis the probability of demands with the same probability of failure is given. There is a (usually small) probability P_A of selecting a demand on which the score function is 1, i.e. on which version A will fail. The majority of demands, with probability $(1 - P_A)$, do not produce a failure. The probability P_A characterises how likely it is for version A to fail on a randomly selected demand. Sometimes, implicitly, Fig. 1a is substituted with Fig. 1b in which *every* demand has the same probability of failure, P_A . The difference between the two is substantial. Although in both cases the mean value of Ω_A is the same, P_A , Fig. 1b misleadingly hides variability, i.e., makes all demands equally likely to produce a failure. With the situation in Fig. 1b, the covariance term in (4) would be 0 and would allow us to use the independence assumption to calculate the probability of system failure for a 1-out-of-2 system. With Fig. 1a, this is not a legal operation.

How useful is expression (4) in practice? We may reasonably expect to be able to estimate individual probabilities of failure (or bounds on them) via operational testing. On the other hand, estimating the covariance term by testing the two versions together is as difficult as estimating the failure probability of the two-version system: very difficult, usually, as design diversity is used in the attempt to satisfy very high reliability requirements [3]. Geist et al. [18] have proposed an *indirect* method. For real-time software, they suggest that the covariance term could be estimated from the covariance between the *execution times* of the versions. We think that such methods depend on many unwarranted assumptions. Rather than

seeking methods for estimating the covariance term, we will study how reliability assessment can make use of the data that may be available from current practice in software reliability engineering.

4. Probability of failure on demand given knowledge on subdomains

Practitioners and researchers of software testing have found it useful to reason in terms of *subdomains* in the demand space of a program. Subdomains are simply subsets of the demand space, defined on the basis of some plausibly useful criterion. For instance, testers find it useful to define subdomains on the basis of which "function" of the program the demands invoke, or on the basis of which parts of the code they cause to be executed. In some other cases the environment in which the system is operating can have different states which affect substantially software reliability. Such states could, for instance, be associated with failures of hardware components, e.g. sensors, [4] and "naturally" partition the demand space.

For our two versions, A and B, we now consider subdomains that form a *partition* in the demand space, i.e., they cover it completely and no two subdomains have elements in common. Formally, the demand space D is divided into a set of subdomains, $\alpha = \{S_1, S_2, \dots, S_n\}$, with:

$$D = S_1 \cup S_2 \cup \dots \cup S_n \text{ and } S_i \cap S_j = \emptyset \text{ for } i \neq j \mid i, j \in \{1, 2, \dots, n\}.$$

Although very little is known about the score functions of the two versions, we may be able to say something about the failure probabilities of the two versions as a function of the subdomain to which a demand belongs.

The typical way of gaining reliability information is by monitoring operation or (statistically) realistic testing. In particular, if two pre-existing products are being combined into two-channel system, there may be extensive experience of previous operation and recorded failures (or lack thereof for highly reliable systems).

The versions are executed on a *randomly selected demand*. For each subdomain S_i ($i=1, \dots, n$), we assume that the following probabilities are known: the probability of a demand being drawn from S_i during software operation, $P(S_i)$, and the probabilities of failure of A and B for demands from S_i , $P(A/S_i)$ and $P(B/S_i)$.

We will be interested in the probability of failure of the 1-out-of-2 system, $P(A, B \text{ fail on } X)$, for a randomly selected X . For the sake of brevity we will write $P(A, B)$ instead of $P(A, B \text{ fail on } X)$ and $P(A, B/S_i)$ instead of $P(A, B \text{ fail on } X \in S_i)$.

For each subdomain, according to (4):

$$P(A, B/S_i) = P(A/S_i) \cdot P(B/S_i) + \text{cov}_i(\Omega_A, \Omega_B).$$

where $\text{cov}_i(\Omega_A, \Omega_B)$ is calculated for demands taken from the subdomain S_i .

The probability of coincident failure on a randomly selected demand is then:

$$\begin{aligned} P(A, B) &= \sum_i P(A, B/S_i) \cdot P(S_i) = \sum_i \left(P(A/S_i) \cdot P(B/S_i) + \text{cov}_i(\Omega_A, \Omega_B) \right) \cdot P(S_i) = \\ &= \sum_i \left(P(A/S_i) \cdot P(B/S_i) \right) \cdot P(S_i) + \sum_i \left(\text{cov}_i(\Omega_A, \Omega_B) \right) \cdot P(S_i) = \\ &= P_{\text{sub-ind}}(A, B) + E(\text{cov}_i(\Omega_A, \Omega_B)) \end{aligned} \quad (5)$$

The first term in the right hand sum of (5) could be called the probability of coincident failure if inside every *subdomain* the versions failed *independently* (hence the index "sub-ind"). The second term is the mean (over the set of subdomains) of the covariance between the score functions of the versions in a randomly chosen subdomain. Obviously, it depends on the versions themselves and the way the demand space has been partitioned. This term does not seem to be directly measurable. $P_{\text{sub-ind}}(A, B)$ however, can be calculated using parameters that we have assumed to be known.

Let us introduce some more notations to facilitate the analysis of $P_{sub-ind}(A, B)$. If we choose a subdomain S^* at random, according to the probability distribution $P(S_i)$, we can define the random variable S_A as the probability that version A fails on a demand selected at random from S^* (according to the demand profile). Then S_A , “the probability of failure of A in a randomly selected subdomain”, takes on the value $P(A/S_i)$ with probability $P(S_i)$. Thus the distribution of S_A is completely defined on the set of subdomains, α . Similarly, let S_B and S_{AB} denote the random variables representing the probability of failure of B and of the coincident failure of A and B in a randomly selected subdomain. Clearly:

$$P_{sub-ind}(A, B) = E(S_{AB}) = E(S_A)E(S_B) + \text{cov}(S_A, S_B) \quad (6)$$

where “ E ” stands for expected value and “ cov ” for covariance. Obviously, $E(S_A) = P_A$ and $E(S_B) = P_B$. Therefore, (6) becomes:

$$P_{sub-ind}(A, B) = P_A \cdot P_B + \text{cov}(S_A, S_B) \quad (7)$$

and (5) becomes:

$$P(A, B) = P_A \cdot P_B + \text{cov}(S_A, S_B) + E(\text{cov}_i(\Omega_A, \Omega_B)) \quad (8)$$

$P_{sub-ind}(A, B)$ represents the probability of system failure if we assume that, within every subdomain, the two versions fail independently. This is an extreme assumption. One might believe it to be approximately true, for instance, if subdomains were defined by grouping together demands that require similar operations of the software. One might then claim that over each subdomain the EL and LM “difficulty function” is approximately constant. Equation (7) says that even this assumption does not allow us to claim failure independence. Equation (7) models the effect of variation (in the probability of failure) over the demand space, similarly to equation (4), except in that the demand space is described in less detail.

The covariance $\text{cov}(S_A, S_B)$ is something new, added to the probability of coincident failure of both versions by splitting the demand space into subdomains. There are 3 cases that can be identified with respect to the value taken by the covariance term:

$\text{cov}(S_A, S_B) = 0$. Apart from other possible cases, this holds if the probability of failure for at least one of the versions does not vary between the subdomains, i.e. the version is believed to be equally reliable in all subdomains. This case seems highly unlikely in real situations.

$\text{cov}(S_A, S_B) > 0$. This is bad news, but seems the most common one to expect in practice under the EL hypothesis, i.e., if we have not found a means of “forcing diversity” which would make the difficulty functions negatively correlated.

$\text{cov}(S_A, S_B) < 0$. This is good news. We may do even better than under the assumption of independence. This is a highly desirable case, but difficult to achieve.

4.1. Lower bound on the probability of common failure

The practical utility of (7) is that we *can actually calculate* $\text{cov}(S_A, S_B)$ if we can estimate $P(A/S_i)$, $P(B/S_i)$ and $P(S_i)$ (for all values of i), and we can thus know exactly which one of cases i) through iii) is true. Knowing the probabilities of failure in subdomains reduces our uncertainty about the system’s probability of failure. Under the (generally optimistic) assumption that versions fail *independently* in each subdomain, we can even compute this latter probability, $P_{sub-ind}(A, B)$. This presumably optimistic estimate can be used by a regulator as a “claim limit” on the reliability that can be claimed for a diverse system in a safety case, and will often be a stricter limit than one based on unconditional independence, $P(A, B) = P_A P_B$.

The analysis we developed with reference to demand subdomains can be applied to a wider range of situations. Whenever we have multiple versions (of software or any other system) which may be used in different conditions of use, and we can make statements on their reliability conditional on these various conditions, the same equations will apply. Different

"conditions of use" could represent, for instance, different regimes of operation of a controlled plant, different modes of operation of the software [4], different loads on the software (in situations like those studied in [19]), different environmental conditions affecting reliability of the hardware. This last interpretation, in which the conditional independence assumption might be considered true, produces the Hughes model for common failure in hardware [5]. The general result is that if those conditions that cause a higher failure rate in one version do so in the other as well, the failures of the two versions are positively correlated.

4.2. Upper bound on the probability of common failure

Clearly, a 1-out-of-2 system is at least as reliable as the more reliable of the two channels. This is true on the whole demand space as well as on each subdomain. We can express, therefore, an upper bound on the probability of system failure as a weighted sum of upper bounds within subdomains:

$$P(A, B) \leq \sum_i \min(P(A|S_i), P(B|S_i))P(S_i) \quad (9)$$

which would be lower than the marginal probability of failure of the better channel, $\min(P(A), P(B))$, iff there is at least one subdomain on which the (marginally) better channel is less reliable than the (marginally) worse channel.

We shall call the right-hand term of (9) $P_{upper}(A, B)$.

4.3. Examples

These two fictional examples illustrate the use of the upper and lower bounds.

Example 1

Table 1

Subdomain	Profile, $P(S_i)$	$P(A S_i)$	$P(B S_i)$	$P(A S_i) \cdot P(B S_i)$	$\min(P(A S_i), P(B S_i))$
S_1	0.2	10^{-2}	10^{-2}	10^{-4}	10^{-2}
S_2	0.3	$2 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$4 \cdot 10^{-7}$	$2 \cdot 10^{-4}$
S_3	0.05	10^{-3}	10^{-3}	10^{-6}	10^{-3}
S_4	0.1	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$2.5 \cdot 10^{-7}$	$5 \cdot 10^{-5}$
S_5	0.35	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$4 \cdot 10^{-6}$	$2 \cdot 10^{-3}$
$P(A) = \sum_i P(A S_i)P(S_i)$			$2.8 \cdot 10^{-3}$		
$P(B) = \sum_i P(B S_i)P(S_i)$			$3.4 \cdot 10^{-3}$		
$cov(S_A, S_B)$			$1.21 \cdot 10^{-5}$		
$P_{sub-ind}(A, B) = 2.2 \cdot 10^{-5}$			$P_{upper}(A, B) = 2.8 \cdot 10^{-3} = \min(P(A), P(B))$		

Example 2

Table 2

Subdomain	Profile, $P(S_i)$	$P(A S_i)$	$P(B S_i)$	$P(A S_i) \cdot P(B S_i)$	$\min(P(A S_i), P(B S_i))$
S_1	0.2	10^{-5}	10^{-1}	10^{-6}	10^{-5}
S_2	0.3	10^{-5}	10^{-1}	10^{-6}	10^{-5}
S_3	0.05	10^{-1}	10^{-5}	10^{-6}	10^{-5}
S_4	0.1	10^{-1}	10^{-5}	10^{-6}	10^{-5}
S_5	0.35	10^{-1}	10^{-5}	10^{-6}	10^{-5}
$P(A) = \sum_i P(A S_i)P(S_i)$			$5.0005 \cdot 10^{-2}$		
$P(B) = \sum_i P(B S_i)P(S_i)$			$5.0005 \cdot 10^{-2}$		
$cov(S_A, S_B)$			$-2.5 \cdot 10^{-5}$		
$P_{sub-ind}(A, B) = 10^{-6}$			$P_{upper}(A, B) = 10^{-5} < \min(P(A), P(B))$		

In the first example the bounds are not tight: the ratio of the two is more than 100. Instead, in the second example the bounds are closer by an order of magnitude. In addition, in the first

example the upper bound simply coincides with the *pdf* of the more reliable channel while in the second example the upper bound is 3 orders of magnitude better than the *pdf* of the two individual channels, which are equally reliable. The gain is clearly due to the negative covariance between S_A and S_B , which in Example 2 has a substantial value (comparable in absolute value with the product of the individual marginal *pdfs*). The channels in the second example are “really failure-diverse”: where the first channel has a modest reliability the second channel is very reliable, and vice-versa.

4.4. Practical use of the upper and lower bounds

In practice, how would these upper and lower bounds be used?

Let us first circumscribe the situations in which they would be useful:

- two software products A and B are intended to be combined in a two-version, 1-out-of-2 system, and for both of them there are previous records of (separate) usage and failures;
- the demand profile is identical, both between the environments where A and B were previously used and the environment in which the new AB system will be used); or, alternatively, the demand profiles conditional on each subdomain are identical, and the probabilities $P(S_i)$ in all three environments are known.

Once satisfied that these conditions apply, an assessor or regulator would, in the typical situation with safety-critical software, try to err on the conservative side. There are three scenarios of interest for using our bounds based on the separate usage-and-failure statistics of the two versions:

- *deriving an upper bound on the probability of joint failure.* The upper bound $P_{upper}(A,B)$ would be taken as an acceptable claim on the *pdf* of the two-version system, to be used, for instance, in probabilistic safety assessment of a plant protected by this two-version system. Since both $P_{upper}(A,B)$ and $\min(P(A),P(B))$ are conservative estimates, the smaller (less pessimistic) of the two would be preferred. The value of $P_{sub-ind}(A,B)$ would be irrelevant;
- *claim limits.* Sometimes, a claim will be made that the true *pdf* of the two-version system is substantially lower than $P_{upper}(A,B)$, for instance by invoking non-statistically based arguments about the quality of the two versions. A regulator may consider such an argument plausible, and yet want some check on its realism. Then, $P_{sub-ind}(A,B)$ becomes useful: if the claimed *pdf* is lower than $P_{sub-ind}(A,B)$, the regulator is justified in doubting it or requiring a very strong argument to accept it. The claim indeed implies claiming that, for demands from individual subdomains, the two versions are (on average over all subdomains) “better than independent”, a very strong claim;
- *precise estimation.* A third, different situation may exist: rather than trying to be conservative, the assessor would like as precise as possible an estimate of the true *pdf* of the two-version system. Then, having both an upper and a lower bound is useful if they happen to be close together.

In the next section, we show examples of calculation of the two bounds, demonstrating some interesting cases. We also show how to account for the fact that in reality we do not know with certainty the probabilities of failures of the versions in each subdomain, but we have to infer them, with some uncertainty, from the available records of usage and failure counts.

5. Example application to experimental data

We can demonstrate the application of the model using the data reported in [20]. In a controlled experiment, 20 independently developed versions were tested under various “states” of the software’s environment. These states, $S_{0,0}$, $S_{0,1}$, $S_{1,0}$, $S_{2,0}$ and $S_{2,1}$, differed in the fact that different subsets of the (redundant) set of sensors read by the software were

faulty prior or during the execution of the test cases. $S_{1,0}$, for instance, denotes the “state” of the environment with a single faulty sensor prior to testing and no more sensor failures during the testing. Details of the experimental setup and meaning of states are given in [20]. The experiment’s results thus include measurements for many version pairs conditional on various “states” of the environment. Each “state” defines in our terms a subdomain in the demand space - the space of possible test runs. We select three version pairs, (4,5), (7,11) and (11,18), to illustrate the derivation of lower and upper bounds on the probability of joint failure (joint *pdf*). None of the five versions considered passed all tests.

An excerpt from the testing results [20] with failure counts of the chosen versions is presented in Table 3. We use the notation “P(4,5)”, for instance, to designate the joint *pdf* of the two versions 4 and 5.

Table 3. Failure counts of selected versions

Version	“States” of the environment [20]					
	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
4	0	2	4	51	9	63
5	0	1	0	0	0	0
7	0	3	0	12360	110	41032
11	14	179	20	155	13	41662
18	53	194	13	55	7	40
Tests	309442	134135	129999	101151	102510	143509

The authors of [20] did not estimate the likelihood of different states of the environment, limiting the discussion to pointing out that “no failure of sensors” will almost always be the case ($P(S_{0,0}) \approx 1$) and therefore the impact of other states of the environment on the marginal probability of failure of versions will be negligible.

The next table shows three hypothetical “demand profiles” which we have used in our calculations, so that we can also illustrate the effect of the demand profile on the bounds we seek.

Table 4. Demand profiles

	OP1	OP2	OP3
$P(S_{0,0})$	0.99	0.4	0.2
$P(S_{0,1})$	0.005	0.2	0.16
$P(S_{1,0})$	0.003	0.2	0.16
$P(S_{1,1})$	0.001	0.1	0.16
$P(S_{2,0})$	0.0005	0.05	0.16
$P(S_{2,1})$	0.0005	0.05	0.16

OP1 represents a more “realistic” profile, where the likelihood that all sensors function correctly is 99%. The likelihood of the other states (with sensor failures) is much lower.

In OP3 the states are almost equally likely, which may be used as a conservative measure to assess the system reliability. OP2 is “in between”.

5.1. Application using point estimates

In this section, we use, for the probabilities of failure of each version in each of the states, the classical unbiased point estimator given by the sample proportion, i.e. the ratio between the number of version failures and the number of tests conducted. For instance, we use for $P(11/S_{0,1})$ the estimated value $179/134135 = 0.001334$ because version 11 has failed 179 times in 134135 tests conducted in state $S_{0,1}$. For these point estimates and all measures derived from them we will use the subscript “*pe*”, e.g., $P_{pe}(A/S_i)$ and $P_{pe}(B/S_i)$, $P_{sub-ind_pe}(A,B)$, $P_{upper_pe}(A,B)$.

The bounds $P_{sub-ind_pe}(A,B)$ and $P_{upper_pe}(A,B)$ and the $cov_{pe}(S_A, S_B)$ under OP1 are shown in Tables 5 - 7 which illustrate the usefulness and limitations of the proposed bounds.

Table 5. Pair (4,5) under profile OP1

States	Profile, $P(S_i)$	$P_{pe}(4/S_i)$	$P_{pe}(5/S_i)$	$P_{pe}(4/S_i) P_{pe}(5/S_i)$	$\min(P_{pe}(4/S_i), P_{pe}(5/S_i))$
$S_{0,0}$	0.99	0	0	0	0
$S_{0,1}$	0.005	$1.49 \cdot 10^{-5}$	$7.46 \cdot 10^{-6}$	$1.11 \cdot 10^{-10}$	$7.46 \cdot 10^{-6}$
$S_{1,0}$	0.003	$3.08 \cdot 10^{-5}$	0	0	0
$S_{1,1}$	0.001	$5.04 \cdot 10^{-4}$	0	0	0
$S_{2,0}$	0.0005	$8.78 \cdot 10^{-5}$	0	0	0
$S_{2,1}$	0.0005	$4.39 \cdot 10^{-4}$	0	0	0
$P_{pe}(4)$				$9.34 \cdot 10^{-7}$	
$P_{pe}(5)$				$3.73 \cdot 10^{-8}$	
$P_{pe}(4)P_{pe}(5)$				$3.48 \cdot 10^{-14}$	
$cov_{pe}(S_4, S_5)$				$5.21 \cdot 10^{-13}$	
$P_{sub-ind_pe}(4,5) = 5.56 \cdot 10^{-13}$		$P_{upper_pe}(4,5) = 3.73 \cdot 10^{-8}$			

Table 6. Pair (7,11) under profile OP1

States	Profile, $P(S_i)$	$P_{pe}(7/S_i)$	$P_{pe}(11/S_i)$	$P_{pe}(7/S_i) P_{pe}(11/S_i)$	$\min(P_{pe}(7/S_i), P_{pe}(11/S_i))$
$S_{0,0}$	0.99	0	$4.52 \cdot 10^{-5}$	0	0
$S_{0,1}$	0.005	$2.24 \cdot 10^{-5}$	$1.33 \cdot 10^{-3}$	$2.98 \cdot 10^{-8}$	$2.24 \cdot 10^{-5}$
$S_{1,0}$	0.003	0	$1.54 \cdot 10^{-4}$	0	0
$S_{1,1}$	0.001	0.122	$1.53 \cdot 10^{-3}$	$1.87 \cdot 10^{-4}$	$1.53 \cdot 10^{-3}$
$S_{2,0}$	0.0005	$1.07 \cdot 10^{-3}$	$1.27 \cdot 10^{-4}$	$1.36 \cdot 10^{-7}$	$1.27 \cdot 10^{-4}$
$S_{2,1}$	0.0005	0.29	0.29	$8.30 \cdot 10^{-2}$	0.29
$P_{pe}(7)$				$2.66 \cdot 10^{-4}$	
$P_{pe}(11)$				$1.99 \cdot 10^{-4}$	
$P_{pe}(7)P_{pe}(11)$				$5.28 \cdot 10^{-8}$	
$cov_{pe}(S_7, S_{11})$				$4.16 \cdot 10^{-5}$	
$P_{sub-ind_pe}(7,11) = 4.17 \cdot 10^{-5}$		$P_{upper_pe}(7,11) = 1.45 \cdot 10^{-4}$			

Table 7. Pair (11,18) under profile OP1

States	Profile, $P(S_i)$	$P_{pe}(11/S_i)$	$P_{pe}(18/S_i)$	$P_{pe}(11/S_i) P_{pe}(18/S_i)$	$\min(P_{pe}(11/S_i), P_{pe}(18/S_i))$
$S_{0,0}$	0.99	$4.52 \cdot 10^{-5}$	$1.71 \cdot 10^{-4}$	$7.75 \cdot 10^{-9}$	$4.52 \cdot 10^{-5}$
$S_{0,1}$	0.005	$1.33 \cdot 10^{-3}$	$1.45 \cdot 10^{-3}$	$1.93 \cdot 10^{-6}$	$1.33 \cdot 10^{-3}$
$S_{1,0}$	0.003	$1.54 \cdot 10^{-4}$	10^{-4}	$1.54 \cdot 10^{-8}$	10^{-4}
$S_{1,1}$	0.001	$1.53 \cdot 10^{-3}$	$5.44 \cdot 10^{-4}$	$8.33 \cdot 10^{-7}$	$5.44 \cdot 10^{-4}$
$S_{2,0}$	0.0005	$1.27 \cdot 10^{-4}$	$6.83 \cdot 10^{-5}$	$8.66 \cdot 10^{-9}$	$6.83 \cdot 10^{-5}$
$S_{2,1}$	0.0005	0.29	$2.79 \cdot 10^{-4}$	$8.09 \cdot 10^{-5}$	$2.79 \cdot 10^{-4}$
$P_{pe}(11)$				$1.99 \cdot 10^{-4}$	
$P_{pe}(18)$				$1.78 \cdot 10^{-4}$	
$P_{pe}(11)P_{pe}(18)$				$3.53 \cdot 10^{-8}$	
$cov_{pe}(S_{11}, S_{18})$				$2.33 \cdot 10^{-8}$	
$P_{sub-ind_pe}(11,18) = 5.87 \cdot 10^{-8}$		$P_{upper_pe}(11,18) = 5.25 \cdot 10^{-5}$			

The three tables reveal the following:

- the lower bound we have identified, $P_{sub-ind_pe}(A,B)$, may differ from the estimate given by the naïve assumption of marginal independence between failures of the two channels. For the first pair, (4,5), the lower bound $P_{sub-ind_pe}(4,5)$ is an order of magnitude greater - $5.56 \cdot 10^{-13}$ vs $3.48 \cdot 10^{-14}$ - than the pdf given by the independence assumption, but is still a very small number. With the second pair, (7,11), due to the strong positive covariance over subdomains, $4.16 \cdot 10^{-5}$, which is comparable with the marginal probabilities of failure of the two channels, $P_{sub-ind_pe}(7,11)$ is almost three orders of magnitude more conservative than the pdf under the assumption of marginal independence: $4.17 \cdot 10^{-5}$ vs $5.28 \cdot 10^{-8}$. Finally, with the third pair we have an example when $P_{sub-ind_pe}(11,18) = 5.87 \cdot 10^{-8}$ is less than twice greater than under the assumption of marginal independence, $3.53 \cdot 10^{-8}$. Thus,

we have examples in which assuming independence may lead to underestimation of the joint pdf . With this demand profile, OP1, we do not have an example of $P_{sub-ind_pe}(A,B)$ being lower than the value calculated under the assumption of marginal independence, which as the theory suggests is possible.

- $P_{upper_pe}(A,B)$ may be better (i.e. lower) than the pdf of the better channel. In the pair (4,5), version 5 is better than version 4 in each subdomain. Therefore, $P_{upper_pe}(4,5)$ equals the pdf of version 5. With the pairs, (7,11) and (11,18), $P_{upper_pe}(A,B)$ is better than the pdf of the more reliable version, versions 11 and 18 respectively. This is due to the fact that in neither pair one of the versions is universally (i.e. in every subdomain) better than the other. The improvement of $P_{upper_pe}(A,B)$ compared with the pdf of the better channel is marginal for the pair (7,11) and substantial (more than 3 times) for the pair (11,18). Notice that for both (7,11) and (11,18), under the demand profile OP1, the covariance term is positive and yet $P_{upper_pe}(A,B)$ is better than the pdf of the better version.
- the bounds may or may not be tight. The pair (4,5) is an example of very loose bounds - $P_{upper_pe}(4,5)$ is several orders of magnitude greater than $P_{sub-ind_pe}(4,5)$. This is not very good news for the assessor of the joint pdf . The uncertainty about this probability should be reduced using other methods or at the expense of further V&V if $P_{upper_pe}(4,5)$ is not satisfactory. The second pair (7,11) is an example of very tight bounds: the ratio of $P_{upper_pe}(7,11)$ and $P_{sub-ind_pe}(7,11)$ is less than 4.

The results shown in Tables 5-7 are under the demand profile OP1. The effect of the demand profile on the bounds is summarised in Table 8.

Table 8. Effect of demand profiles on the joint $pdfs$

Pair		$P_{pe}(4)$	$P_{pe}(5)$	$cov_{pe}(S_A, S_B)$	$P_{pe}(4)P_{pe}(5)$	$\min(P_{pe}(4), P_{pe}(5))$	$P_{sub-ind_pe}(4,5)$	$P_{upper_pe}(4,5)$
(4,5)	OP1	$9.34 \cdot 10^{-7}$	$3.73 \cdot 10^{-8}$	$5.21 \cdot 10^{-13}$	$3.48 \cdot 10^{-14}$	$3.73 \cdot 10^{-8}$	$5.56 \cdot 10^{-13}$	$3.73 \cdot 10^{-8}$
	OP2	$8.59 \cdot 10^{-5}$	$1.49 \cdot 10^{-6}$	$-1.1 \cdot 10^{-10}$	$1.28 \cdot 10^{-10}$	$1.49 \cdot 10^{-6}$	$2.22 \cdot 10^{-11}$	$1.49 \cdot 10^{-6}$
	OP3	$1.72 \cdot 10^{-4}$	$1.19 \cdot 10^{-6}$	$-1.9 \cdot 10^{-10}$	$2.05 \cdot 10^{-10}$	$1.19 \cdot 10^{-6}$	$1.78 \cdot 10^{-11}$	$1.19 \cdot 10^{-6}$
(7,11)		$P_{pe}(7)$	$P_{pe}(11)$	$cov_{pe}(S_A, S_B)$	$P_{pe}(7)P_{pe}(11)$	$\min(P_{pe}(7), P_{pe}(11))$	$P_{sub-ind_pe}(7,11)$	$P_{upper_pe}(7,11)$
	OP1	$2.66 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$	$4.16 \cdot 10^{-5}$	$5.28 \cdot 10^{-8}$	$1.99 \cdot 10^{-4}$	$4.17 \cdot 10^{-5}$	$1.45 \cdot 10^{-4}$
	OP2	$2.66 \cdot 10^{-2}$	$1.50 \cdot 10^{-2}$	$3.77 \cdot 10^{-3}$	$3.98 \cdot 10^{-4}$	$1.50 \cdot 10^{-2}$	$4.17 \cdot 10^{-3}$	$1.45 \cdot 10^{-2}$
	OP3	$6.55 \cdot 10^{-2}$	$4.70 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	$3.08 \cdot 10^{-3}$	$4.70 \cdot 10^{-2}$	$1.33 \cdot 10^{-2}$	$4.60 \cdot 10^{-2}$
(11,18)		$P_{pe}(11)$	$P_{pe}(18)$	$cov_{pe}(S_A, S_B)$	$P_{pe}(11)P_{pe}(18)$	$\min(P_{pe}(11), P_{pe}(18))$	$P_{sub-ind_pe}(11,18)$	$P_{upper_pe}(11,18)$
	OP1	$1.99 \cdot 10^{-4}$	$1.78 \cdot 10^{-4}$	$2.33 \cdot 10^{-8}$	$3.53 \cdot 10^{-8}$	$1.78 \cdot 10^{-4}$	$5.87 \cdot 10^{-8}$	$5.25 \cdot 10^{-5}$
	OP2	$1.50 \cdot 10^{-2}$	$4.49 \cdot 10^{-4}$	$-2.2 \cdot 10^{-6}$	$6.74 \cdot 10^{-6}$	$4.49 \cdot 10^{-4}$	$4.52 \cdot 10^{-6}$	$3.77 \cdot 10^{-4}$
	OP3	$4.70 \cdot 10^{-2}$	$4.24 \cdot 10^{-4}$	$-6.5 \cdot 10^{-6}$	$1.99 \cdot 10^{-5}$	$4.24 \cdot 10^{-4}$	$1.34 \cdot 10^{-5}$	$3.81 \cdot 10^{-4}$

The demand profile does affect the probability of failure of both versions and pairs. We see dramatic change of the bounds. With the first pair, both $P_{sub-ind_pe}(4,5)$ and $P_{upper_pe}(4,5)$ vary together, by almost two orders of magnitude. Thus the bounds remain loose for all three demand profiles. This is due to the fact that version 5 failed in a single subdomain and therefore both bounds are proportional to the probability of a demand from that single subdomain.

The demand profile changes the bounds on the joint pdf of pairs (7,11) and (11,18) too. The effect of the profile on the tightness of the bounds is different, however, for these two pairs. The bounds for (7,11) remain tight: the ratio of $P_{upper_pe}(7,11)$ and $P_{sub-ind_pe}(7,11)$ stays approximately 3.5 despite the fact that the bounds change by two orders of magnitude. With the pair (11,18), the effect of the demand profile on the bounds is different: while under OP1 the bounds are very loose under OP3 they become very tight (OP2 being in between with bounds moderately tight).

The third pair, (11,18), is particularly interesting because under OP2 and OP3 the covariance term becomes negative and $P_{sub-ind_pe}(11,18)$ becomes lower than the product $P_{pe}(11)P_{pe}(18)$. Thus, assuming marginal independence between failures is not necessarily an optimistic assumption.

5.2. Application using confidence bounds

We have used point estimates of the various probabilities of failure in subdomains, $P(A/S_i)$, $P(B/S_i)$, to derive a point estimate for the joint *pdf* of a 2-version system. Point estimates may be unsatisfactory if there is great uncertainty about the true values of the estimated measures. So, we now apply the same procedure as in Section 5.1, but using one-sided confidence bounds on the probabilities of failure of each version in each of the subdomains. For each subdomain, we compute Bayesian bounds on the *pdf* of versions based on uniform priors, as described in [3], [21]³. We use the notation $P_{99\%}(A/S_i)$ for a 99% confidence *upper* bound (i.e., such that the probability of the true measure being smaller is 99%), and $P_{1\%}(A/S_i)$ for a 99% confidence *lower* bound (i.e., such that the probability of the true measure being smaller is 1%). The derived measures based on these bounds will be denoted with the same subscripts, e.g. $P_{sub-ind99\%}(A,B)$.

Table 9. Upper bounds with 99% confidence

State, S_i	$P_{99\%}(4/S_i)$	$P_{99\%}(5/S_i)$	$P_{99\%}(7/S_i)$	$P_{99\%}(11/S_i)$	$P_{99\%}(18/S_i)$
$S_{0,0}$	$1.49 \cdot 10^{-5}$	$1.49 \cdot 10^{-5}$	$1.49 \cdot 10^{-5}$	$8.22 \cdot 10^{-5}$	0.000234
$S_{0,1}$	$6.27 \cdot 10^{-5}$	$4.95 \cdot 10^{-5}$	$7.49 \cdot 10^{-5}$	0.001585	0.001707
$S_{1,0}$	$8.93 \cdot 10^{-5}$	$3.54 \cdot 10^{-5}$	$3.54 \cdot 10^{-5}$	0.000255	0.000186
$S_{1,1}$	0.000694	$4.55 \cdot 10^{-5}$	0.124608	0.001844	0.00074
$S_{2,0}$	0.000183	$4.49 \cdot 10^{-5}$	0.001336	0.000235	0.000156
$S_{2,1}$	0.000586	$3.21 \cdot 10^{-5}$	0.288701	0.293104	0.0004

The 99% Bayesian bound is substantially more conservative than the point estimate in cases with zero or few observed failures (Table 10 shows some examples). The two are close together, instead, if many failures were observed (this happens for instance with versions 11 and 18, in every subdomain).

Table 10. Version 5: estimates of the conditional *pdf*, $P(5/S_i)$.

	$P(5/S_{0,0})$	$P(5/S_{0,1})$	$P(5/S_{1,0})$	$P(5/S_{1,1})$	$P(5/S_{2,0})$	$P(5/S_{2,1})$
point estimate, $P_{pe}(5/S_i)$	0	$7.46 \cdot 10^{-6}$	0	0	0	0
99% bound, $P_{99\%}(5/S_i)$	$1.49 \cdot 10^{-5}$	$4.95 \cdot 10^{-5}$	$3.54 \cdot 10^{-5}$	$4.55 \cdot 10^{-5}$	$4.49 \cdot 10^{-5}$	$3.21 \cdot 10^{-5}$

In Section 4.4, we identified three possible situations for an assessor:

- the assessor seeks an upper bound, based on the failure statistics of the two versions, for their probability of joint failure. Then he/she can use the Bayesian 99% confidence bounds, $P_{99\%}(A/S_i)$ and $P_{99\%}(B/S_i)$, and calculate $P_{upper99\%}(A,B)$ ⁴.

Table 11 shows the calculation results, and compares $P_{upper99\%}(A,B)$ with $P_{upper_pe}(A,B)$ (last two columns). As expected, the former is more conservative, and substantially so for the pair (4,5), for which few failures were observed.

³ We apply the inference procedure separately to the recorded behaviour of each version, to derive the version's *pdf*. In some situations, this would not be enough: failures (resp. proper operation) of one version would be evidence about the "difficulty" of the function required, and thus affect our expectations about the reliability of the *other* version. With the Bayesian approach, this more complex analysis is feasible; however it is beyond the scope of this paper.

⁴ The probability of $P_{upper99\%}(A,B)$ being greater than the true value of $P_{upper}(A,B)$ could only be computed knowing the joint distributions of all the 12 parameters $P(A/S_i)$, $P(B/S_i)$. A conservative statement, though, is that it is at least 88%, since 12% is the maximum possible probability of at least one of the 12 parameters being greater than the corresponding 99% Bayesian confidence bound, which in turn is a necessary condition for $P_{upper99\%}(A,B)$ to be greater than the true $P_{upper}(A,B)$.

Table 11. Upper bounds on joint pfd using Bayesian upper bounds on version pfd s

Pair		$P_{99\%}(4)$	$P_{99\%}(5)$	$\min(P_{99\%}(4), P_{99\%}(5))$	$P_{upper99\%}(4,5)$	$P_{upper_pe}(4,5)$
(4,5)	OP1	$1.64 \cdot 10^{-5}$	$1.52 \cdot 10^{-5}$	$1.52 \cdot 10^{-5}$	$1.52 \cdot 10^{-5}$	$3.73 \cdot 10^{-8}$
	OP2	$1.44 \cdot 10^{-4}$	$3.13 \cdot 10^{-5}$	$3.13 \cdot 10^{-5}$	$3.13 \cdot 10^{-5}$	$1.49 \cdot 10^{-6}$
	OP3	$2.61 \cdot 10^{-4}$	$3.62 \cdot 10^{-5}$	$3.62 \cdot 10^{-5}$	$3.62 \cdot 10^{-5}$	$1.19 \cdot 10^{-6}$
(7,11)		$P_{99\%}(7)$	$P_{99\%}(11)$	$\min(P_{99\%}(7), P_{99\%}(11))$	$P_{upper99\%}(7,11)$	$P_{upper_pe}(7,11)$
	OP1	$2.85 \cdot 10^{-4}$	$2.39 \cdot 10^{-4}$	$2.39 \cdot 10^{-4}$	$1.62 \cdot 10^{-4}$	$1.45 \cdot 10^{-4}$
	OP2	$2.70 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	$1.47 \cdot 10^{-2}$	$1.45 \cdot 10^{-2}$
	OP3	$6.63 \cdot 10^{-2}$	$4.76 \cdot 10^{-2}$	$4.76 \cdot 10^{-2}$	$4.65 \cdot 10^{-2}$	$4.60 \cdot 10^{-2}$
(11,18)		$P_{99\%}(11)$	$P_{99\%}(18)$	$\min(P_{99\%}(11), P_{99\%}(18))$	$P_{upper99\%}(11,18)$	$P_{upper_pe}(11,18)$
	OP1	$2.39 \cdot 10^{-4}$	$2.42 \cdot 10^{-4}$	$2.39 \cdot 10^{-4}$	$9.09 \cdot 10^{-5}$	$5.25 \cdot 10^{-5}$
	OP2	$1.53 \cdot 10^{-2}$	$5.74 \cdot 10^{-4}$	$5.74 \cdot 10^{-4}$	$4.89 \cdot 10^{-4}$	$3.77 \cdot 10^{-4}$
	OP3	$4.76 \cdot 10^{-2}$	$5.57 \cdot 10^{-4}$	$5.57 \cdot 10^{-4}$	$5.07 \cdot 10^{-4}$	$3.81 \cdot 10^{-4}$

- the assessor wishes to use the statistics to obtain a *lower* bound on the joint pfd , to use as a *claim limit*. If, for instance, the assessor computes a 99% lower bound on the true $P(A/S_i)$ and $P(B/S_i)$, and from these derives an estimate $P_{sub-ind1\%}(A,B)$, with probability at least 0.88 the calculated $P_{sub-ind1\%}(A,B)$ will be lower than the “true” lower bound $P_{sub-ind}(A,B)$. This will give strong reasons for rejecting any claim of joint pfd lower than the calculated $P_{sub-ind1\%}(A,B)$.

Table 12 shows the 99% lower bounds on the conditional pfd s of the chosen versions, $P_{1\%}(A/S_i)$.

Table 12. Lower bounds with 99% confidence

State, S_i	$P_{1\%}(4/S_i)$	$P_{1\%}(5/S_i)$	$P_{1\%}(7/S_i)$	$P_{1\%}(11/S_i)$	$P_{1\%}(18/S_i)$
$S_{0,0}$	$3.25 \cdot 10^{-8}$	$3.25 \cdot 10^{-8}$	$3.25 \cdot 10^{-8}$	$2.42 \cdot 10^{-5}$	0.000124
$S_{0,1}$	$3.25 \cdot 10^{-6}$	$1.11 \cdot 10^{-6}$	$6.14 \cdot 10^{-6}$	0.00112	0.001223
$S_{1,0}$	$9.84 \cdot 10^{-6}$	$7.73 \cdot 10^{-8}$	$7.73 \cdot 10^{-8}$	$9.1 \cdot 10^{-5}$	$5.22 \cdot 10^{-5}$
$S_{1,1}$	0.000363	$9.94 \cdot 10^{-8}$	$7.73 \cdot 10^{-8}$	0.00127	0.000396
$S_{2,0}$	$4.03 \cdot 10^{-5}$	$9.8 \cdot 10^{-8}$	0.000858	$6.62 \cdot 10^{-5}$	$2.83 \cdot 10^{-5}$
$S_{2,1}$	0.000327	$7 \cdot 10^{-8}$	0.283152	0.287529	0.000192

The calculated $P_{sub-ind1\%}(A,B)$ for the selected pairs of versions are shown in Table 13 together with the lower bounds $P_{sub-ind_pe}(A,B)$.

Table 13. Lower bounds on joint pfd using Bayesian lower bounds on version pfd s

Pair		$P_{1\%}(4)$	$P_{1\%}(5)$	$cov_{1\%}(S_A, S_B)$	$P_{1\%}(4)P_{1\%}(5)$	$P_{sub-ind1\%}(4,5)$	$P_{sub-ind_pe}(4,5)$
(4,5)	OP1	$6.24 \cdot 10^{-7}$	$3.81 \cdot 10^{-8}$	$4.7 \cdot 10^{-14}$	$2.38 \cdot 10^{-14}$	$7.08 \cdot 10^{-14}$	$5.56 \cdot 10^{-13}$
	OP2	$5.73 \cdot 10^{-5}$	$2.68 \cdot 10^{-7}$	$-9.5 \cdot 10^{-12}$	$1.54 \cdot 10^{-11}$	$5.82 \cdot 10^{-12}$	$2.22 \cdot 10^{-11}$
	OP3	$1.19 \cdot 10^{-4}$	$2.39 \cdot 10^{-7}$	$-1.8 \cdot 10^{-11}$	$2.84 \cdot 10^{-11}$	$1.08 \cdot 10^{-11}$	$1.78 \cdot 10^{-11}$
(7,11)		$P_{1\%}(7)$	$P_{1\%}(11)$	$cov_{1\%}(S_A, S_B)$	$P_{1\%}(7)P_{1\%}(11)$	$P_{sub-ind1\%}(7,11)$	$P_{sub-ind_pe}(7,11)$
	OP1	$2.62 \cdot 10^{-4}$	$1.75 \cdot 10^{-4}$	$4.08 \cdot 10^{-5}$	$4.58 \cdot 10^{-8}$	$4.09 \cdot 10^{-5}$	$4.17 \cdot 10^{-5}$
	OP2	$2.62 \cdot 10^{-2}$	$1.48 \cdot 10^{-2}$	0.0037	$3.86 \cdot 10^{-4}$	$4.09 \cdot 10^{-3}$	$4.17 \cdot 10^{-3}$
	OP3	$6.46 \cdot 10^{-2}$	$4.64 \cdot 10^{-2}$	$1.01 \cdot 10^{-2}$	$3.00 \cdot 10^{-3}$	$1.31 \cdot 10^{-2}$	$1.33 \cdot 10^{-2}$
(11,18)		$P_{1\%}(11)$	$P_{1\%}(18)$	$cov_{1\%}(S_A, S_B)$	$P_{1\%}(11)P_{1\%}(18)$	$P_{sub-ind1\%}(11,18)$	$P_{sub-ind_pe}(11,18)$
	OP1	$1.75 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$1.53 \cdot 10^{-8}$	$2.27 \cdot 10^{-8}$	$3.8 \cdot 10^{-8}$	$5.87 \cdot 10^{-8}$
	OP2	$1.48 \cdot 10^{-2}$	$3.55 \cdot 10^{-4}$	$3.9 \cdot 10^{-6}$	$5.24 \cdot 10^{-6}$	$9.15 \cdot 10^{-6}$	$4.521 \cdot 10^{-6}$
	OP3	$4.64 \cdot 10^{-2}$	$3.27 \cdot 10^{-4}$	$-6.1 \cdot 10^{-6}$	$1.52 \cdot 10^{-5}$	$9.15 \cdot 10^{-6}$	$1.34 \cdot 10^{-5}$

Not surprisingly, the lower bounds $P_{sub-ind1\%}(A,B)$ are more optimistic than the corresponding $P_{sub-ind_pe}(A,B)$.

- last, the assessor may seek tight bounds on the true probability of joint failure. Table 14 compares the intervals obtained with the 99% upper and lower Bayesian confidence bounds with those obtained from the point estimates in the previous section.

Table 14. Intervals for joint *pdf*: point estimates vs. Bayesian bounds

		Intervals based on point estimates		Intervals based on Bayesian bounds	
Pair	OP	$P_{sub-ind\ pe}(4,5)$	$P_{upper\ pe}(4,5)$	$P_{sub-ind1\%}(4,5)$	$P_{upper99\%}(4,5)$
(4,5)	OP1	$5.56 \cdot 10^{-13}$	$3.73 \cdot 10^{-8}$	$7.08 \cdot 10^{-14}$	$1.52 \cdot 10^{-5}$
	OP2	$2.22 \cdot 10^{-11}$	$1.49 \cdot 10^{-6}$	$5.82 \cdot 10^{-12}$	$3.13 \cdot 10^{-5}$
	OP3	$1.78 \cdot 10^{-11}$	$1.19 \cdot 10^{-6}$	$1.08 \cdot 10^{-11}$	$3.62 \cdot 10^{-5}$
(7,11)		$P_{sub-ind\ pe}(7,11)$	$P_{upper\ pe}(7,11)$	$P_{sub-ind1\%}(7,11)$	$P_{upper99\%}(7,11)$
	OP1	$4.17 \cdot 10^{-5}$	$1.45 \cdot 10^{-4}$	$4.09 \cdot 10^{-5}$	$1.62 \cdot 10^{-4}$
	OP2	$4.17 \cdot 10^{-3}$	$1.45 \cdot 10^{-2}$	$4.09 \cdot 10^{-3}$	$1.47 \cdot 10^{-2}$
	OP3	$1.33 \cdot 10^{-2}$	$4.60 \cdot 10^{-2}$	$1.31 \cdot 10^{-2}$	$4.65 \cdot 10^{-2}$
(11,18)		$P_{sub-ind\ pe}(11,18)$	$P_{upper\ pe}(11,18)$	$P_{sub-ind1\%}(11,18)$	$P_{upper99\%}(11,18)$
	OP1	$5.87 \cdot 10^{-8}$	$5.25 \cdot 10^{-5}$	$3.8 \cdot 10^{-8}$	$9.09 \cdot 10^{-5}$
	OP2	$4.52 \cdot 10^{-6}$	$3.77 \cdot 10^{-4}$	$9.15 \cdot 10^{-6}$	$4.89 \cdot 10^{-4}$
	OP3	$1.34 \cdot 10^{-5}$	$3.81 \cdot 10^{-4}$	$9.15 \cdot 10^{-6}$	$5.07 \cdot 10^{-4}$

The intervals computed from the Bayesian bounds are clearly wider. Again, this is much more pronounced for the pair (4,5): having observed fewer failures means greater uncertainty on each of the estimates for $P(4/S_i)$ and $P(5/S_i)$.

Note that assessors may well require different confidence levels from the calculations used in the three cases, although we have arbitrarily used the same level (99%) for all three. With critical software, a conservative bias is usually desired. So, in the first scenario it would make sense to require high confidence levels (a high probability that the statistically calculated bounds are actually worse than the true values). In the second scenario, on the other hand, a calculation showing $P_{sub-ind}$ to be greater than the claimed *pdf* with a probability of, say, 70% or even just 50% would often be enough to prompt a close re-examination of the arguments supporting the claim.

6. Discussion

We have studied models of failure behaviour of two-version software systems, essentially extending the “varying difficulty” assumption first introduced by Eckhardt and Lee. Our main concern has been to help with the problem of evaluating a specific multiple-version system, as opposed to an “average” system. We have modified the previously published models to use the actual failure points in the produced versions rather than the “difficulty function” which describes the probability that a demand *could* be a failure point. We have used parameters describing aggregates of demands, rather than individual demands. The resulting model has both some value in terms of insight into the problem and (which is new) practical applicability: we have shown how to state lower and upper bounds on system *pdf* for a specific two-version system.

We will first discuss the mathematical aspects of our results, and then their practical uses. In mathematical terms, we notice that there is no substantial difference between models based on subdomains and the previous (EL and LM) models based on individual demands. The only difference is in the level of detail with which we model the demand space. The practical difference, of course, is that there is no hope of practical use for individual-point models, but there is for our subdomain- or mode-based models.

Our model operates at a coarser level of presentation than the EL and LM ones. We do not know details about the versions’ behaviours on individual demands inside each subdomain. The correlation between failures of two versions on demands from a certain subdomain (“intra-subdomain” dependence) can take *any value*: zero, positive or negative. The EL model suggests that intra-subdomain *independence* is an optimistic assumption. On this basis, one can establish a lower bound on the probability of system failure. Even if the intra-subdomain dependence varies between subdomains, what matters is the mean over the

subdomains of a measure of this dependence, the last term in equations (5) and (8). It is plausible that this will be positive, so that (9) gives a true lower bound on the system *pfd*, but if it is not we would err on the conservative side - we are not overestimating the reliability of the system.

From the viewpoint of practical application, we have discussed various cases in Section 4.4. Our bounds are only useful if they are substantially narrower than those obtainable by other means. In this sense, the lower bound is useful (i.e., higher than alternative known lower bounds) if the above-mentioned mean measure of intra-domain dependence can be assumed positive. This would be the reasonable position for a conservative assessor.

Our upper bound based on subdomains may not be useful, if it coincides with the *pfd* of the more reliable version, ($\min(P(A), P(B))$). However, in some cases it is substantially smaller than ($\min(P(A), P(B))$), and therefore very useful. These cases cannot be predicted in advance of the actual measurements: we cannot use this upper bound in the safety case for a two-channel system that is yet to be built; but if we are going to build a two-channel system from two pre-existing systems for which (separate) operational experience already exists, this upper bound may be very useful, under the conditions listed in Section 4.4 on the similarity between the old and new operating environments. If the two versions were very reliable (no failures observed), the only reasonable *pfd* estimates for the versions would be upper bounds. These can be used as point estimates, which is questionable, but one can also exploit the possibility of deriving a pessimistic upper bound for system *pfd* as shown in section 5.2.

It is also worth noting that one can also apply our method from section 4 with estimates (of the conditional probabilities of failure of versions in subdomains) which are not based on statistical testing but on other evidence, e.g. from inspection or formal proof.

In conclusion, our results do improve the ability of assessors and regulators to assess a particular diverse-channel system using information specific to that system.

We see two main questions which remain open.

First, in practice we usually deal with very reliable versions (only a few failures are likely to be detected during the testing) and thus inference about the *pfd* of channels is difficult. The natural approach is Bayesian, which allows us to consider explicitly dependencies between one's uncertainties about the *pfd*s of the two versions: failures (respectively proper operation) of one version would affect our expectations about the reliability of the other version.

The other problem, common to all cases of re-use of software with an existing record of operation, is how to take account of differences between the demand profiles under which the previous statistics were collected and the demand profile in the intended operational environment.

Acknowledgement

This work has been partially supported by the following projects on design diversity: DISCS (Diversity In Safety Critical Software), funded by the U.K. Engineering and Physical Sciences Research Council, grant GR/L07673, and DISPO (Diverse Software PrOject), funded by British Energy Generation Ltd under Contract No. PP/79405/MB.

The authors are indebted to Bev Littlewood and Bob Jennings for their comments on previous versions of this article.

References

- [1] M. R. Lyu (Ed.), "Software Fault Tolerance", Wiley, 1995.
- [2] "MoD 00-55 Def Stan 00-55, Requirements for Safety Related Software in Defence Equipment", U.K. Ministry of Defence Issue 2, 1997, 1997.
- [3] B. Littlewood and L. Strigini, "Validation of Ultra-High Dependability for Software-based Systems", Communications of the ACM, 36, pp. 69-80, 1993.

- [4] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors", *IEEE Transactions on Software Engineering*, SE-11, pp. 1511-1517, 1985.
- [5] R. P. Hughes, "A New Approach to Common Cause Failure", *Reliability Engineering*, 17, pp. 211-236, 1987.
- [6] B. Littlewood, "The impact of diversity upon common mode failures", *Reliability Engineering and System Safety*, 51, pp. 101-113, 1996.
- [7] P. G. Bishop, "Software Fault Tolerance by design diversity", in M. Lyu (Ed.) "Software Fault Tolerance", John Wiley & Sons, 1995, pp. 211-229.
- [8] B. Littlewood and D. R. Miller, "Conceptual Modelling of Coincident Failures in Multi-Version Software", *IEEE Transactions on Software Engineering*, SE-15, pp. 1596-1614, 1989.
- [9] V. F. Nicola and A. Goyal, "Modeling of Correlated Failures and Community Error Recovery in Multiversion Software", *IEEE Transactions on Software Engineering*, 16, pp. 350-359, 1990.
- [10] J. C. Knight and N. G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming", *IEEE Transactions on Software Engineering*, SE-12, pp. 96-109, 1986.
- [11] K. B. Djambazov and P. Popov, "The effects of testing on the reliability of single version and 1-out-of-2 software", in *Proc. 6th Int. Symposium on Software Reliability Engineering, ISSRE'95, Toulouse, 1995*, pp. 219-228.
- [12] A. Gnarrarov, J. Arlat and A. Avizienis, "On the Performance of Software Fault-Tolerance Strategies", in *Proc. 10th International Symposium on Fault-Tolerant Computing, Kyoto, Japan, 1980*, pp. 251-253.
- [13] J. Arlat, K. Kanoun and J. C. Laprie, "Dependability Modelling and Evaluation of Software Fault-Tolerant Systems", *IEEE Transactions on Computers*, C-39, pp. 504-513, 1990.
- [14] A. T. Tai, A. Avizienis and J. F. Meyer, "Performability Enhancement of Fault-Tolerant Software", *IEEE Transactions on Reliability*, Special Issue on Fault-Tolerant Software, R-42, pp. 227-237, 1993.
- [15] A. Csenki, "Recovery Block Reliability Analysis with Failure Clustering", in A. Avizienis and J.-C. Laprie (Ed.) "Dependable Computing for Critical Applications (DCCA-1, Santa Barbara, CA, USA, August 1989)", Vienna, Austria, Springer-Verlag, 1991, pp. 75-103.
- [16] J. B. Dugan and M. R. Lyu, "System Reliability Analysis of An N-Version Programming Application", *IEEE Transactions on Reliability*, 43, pp. 513-519, 1994.
- [17] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico and L. Strigini, "A Contribution to the Evaluation of the Reliability of Iterative-Execution Software", *Software Testing, Verification and Reliability*, pp. to appear, 1999.
- [18] R. Geist, A. J. Offutt and F. C. Harris, "Estimation and Enhancement of Real-Time Software Reliability through Mutation Analysis", *IEEE TC*, C-41, pp. 550-558, 1992.
- [19] R. K. Iyer and I. Lee, "Measurement-Based Analysis of Software Reliability", in M. Lyu (Ed.) "Handbook of Software Reliability Engineering", McGraw Hill, 1996, pp. 303-358.
- [20] D. E. Eckhardt, A. K. Caglayan, J. C. Knight, L. D. Lee, D. F. McAllister, M. A. Vouk and J. P. J. Kelly, "An experimental evaluation of software redundancy as a strategy for improving reliability", *IEEE Transactions on Software Engineering*, 17, pp. 692-702, 1991.
- [21] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nicol, B. W. Murrill and J. M. Voas, "Estimating the Probability of Failure When Testing Reveals No Failures", *IEEE Transactions on Software Engineering*, 18, pp. 33-43, 1992.