Reasoning about the Reliability Of Diverse Two-Channel Systems In which One Channel is "Possibly Perfect"

Bev Littlewood
Centre for Software Reliability
City University, London EC1V 0HB, UK

John Rushby
Computer Science Laboratory
SRI International, Menlo Park CA 94025, USA

December 8, 2010

Abstract

This paper refines and extends an earlier one by the first author [17]. It considers the problem of reasoning about the reliability of fault-tolerant systems with two "channels" (i.e., components) of which one, A, because it is conventionally engineered and presumed to contain faults, supports only a claim of reliability, while the other, B, by virtue of extreme simplicity and extensive analysis, supports a plausible claim of "perfection."

We begin with the case where either channel can bring the system to a safe state. The reasoning about system probability of failure on demand (*pfd*) is divided into two steps. The first concerns *aleatory uncertainty* about (i) whether channel A will fail on a randomly selected demand and (ii) whether channel B is imperfect. It is shown that, conditional upon knowing p_A (the probability that A fails on a randomly selected demand) and p_B (the probability that channel B is imperfect), a conservative bound on the probability that the system fails on a randomly selected demand is simply $p_A \times p_B$. That is, there is conditional independence between the events "A fails" and "B is imperfect." The second step of the reasoning involves *epistemic uncertainty* represented by assessors' beliefs about the distribution of $(p_A,\ p_B)$ and it is here that dependence may arise. However, we show that under quite plausible assumptions, a conservative bound on system *pfd* can be constructed from point estimates for just three parameters. We discuss the feasibility of establishing credible estimates for these parameters.

We extend our analysis from faults of omission to those of commission, and then combine these to yield an analysis for monitored architectures of a kind proposed for aircraft.

1 Background

This paper is about the assessment of dependability for fault tolerant software-based systems that employ two design-diverse channels. This type of system architecture is mainly used to provide protection against design faults in safety-critical applications. The paper clarifies and greatly extends a note by one of the authors from a few years ago on a specific instance of the topic [17].

The use of intellectual diversity to improve the dependability of processes is ubiquitous in human activity: witness the old saws "Two heads are better than one" and "Belt *and* braces" (or suspenders for North American readers). We all believe that having

a colleague review our work—"two heads"—is better than conducting the review ourselves, and we generally consider it prudent to have a backup for complicated machines or plans.

The use of diversity to build reliable systems is widespread in non-software-based engineering. For example, in the nuclear industry it is common to have multi-channel protection systems that are based on physically different process variables (e.g., temperature, pressure, flow-rates). The simple idea underlying these applications of diversity is that, whilst it is inevitable that humans make mistakes in designing and building systems, and that these mistakes can cause failures during operation, forcing the systems to be built differently might make the failure processes of different versions be diverse also. In other words, you might expect that even if channel A fails on a particular demand, there is a good chance that channel B will not fail on that demand.

These ideas have been applied to software systems for nearly thirty years, but there has been some controversy about the benefits that the approach brings. On the one hand, there are reports of successful industrial applications of software design diversity: for example, the safety-critical flight control systems of the Airbus fleets [34] have now experienced massive operational exposure [6] with apparently no critical failure (though see [2], which led to modifications in flight software). On the other hand, successive carefully conducted experiments—see e.g., [9, 15]—have shown that multiple diverse channels cannot be assumed to fail independently. Thus, if the two channels of a system each have a probability of failure on demand (*pfd*) of 10^{-3}, it would be *prime facie* wrong to claim a *pfd* of 10^{-6} for the system. In fact, there will generally be positive dependence of failures between the two channels, and so the *pfd* will be greater than 10^{-6}.

Some insight into these experimental results came from probabilistic models of diversity [10,16]. This work gave careful attention to the different notions of "independence" that had hitherto been treated somewhat informally, and showed why versions that had been *developed* "independently" could not necessarily be expected to *fail* "independently." The key idea in this work is a notion of "difficulty" variation over the demand space—informally, some demands are intrinsically more difficult to handle correctly than others. Thus, if channel A fails on a randomly selected demand, this increases the likelihood that the demand is a "difficult" one and hence increases the likelihood that B also will fail. That is, the conditional probability of failure of B, given that A has failed, is greater than its marginal probability of failure, p_B, and the probability of system failure will be greater than the product of the marginal probabilities $p_A \times p_B$.

The picture that emerged from this evidence—industrial experience, experiments, and theoretical modeling—has sometimes been taken to undermine claims for the efficacy of software diversity. Whilst the industrial evidence is positive, it inevitably emerges only after years of operating experience, and even then does not easily allow quantitative claims for achieved reliability. The experimental and modeling evidence, on the other hand, has been taken to be negative.

In fact, this negative view seems somewhat unfair. Although the experiments and models show that assumptions of independence of failures cannot be trusted, this does not mean that there cannot be useful benefits to be gained from design diversity. Ruling out independence means that we cannot simply multiply the *pfd*s of the separate channels to obtain the system *pfd*, but there might still be considerable reliability benefits. The modeling results of Littlewood and Miller [16] actually show that there *will* be benefits under quite plausible assumptions, but they do not help quantify these benefits for

particular instances; the experiments of Knight and Leveson [14] show that the benefits can be considerable *on average*, but again their results do not help to assess a particular system's reliability.

The position for several years, then, has been this: there is ample evidence that design diversity can "work," in some average sense, in delivering improved reliability, but there is very little evidence as to *how much* improvement it delivers even on average,[1] and the important problem of how to estimate the reliability of a particular design-diverse system remains very difficult.

This paper concerns this last problem, and it begins with a rather special class of architectures that appear to bring a simplification to the assessment problem. This work was prompted by one author's involvement in the early 1990s with the difficulties of assessing the reliability of the protection system of the UK's Sizewell B nuclear plant. This system comprises the Primary Protection System (PPS), which is software-based, and the Secondary Protection System (SPS), which is hard-wired, in a "one-out-of-two" (sometimes written 1oo2) configuration—that is, if either of the systems requests it, the reactor "trips" and enters a safe state.

The PPS provides quite extensive and desirable functionality, over and above that required for a simple reactor trip; for example, it has extensive built-in hardware self-testing and redundancy management features to maximize its availability. The result is that, for a safety-critical system, it is quite complex with about 100,000 lines of code. For this reason it was not thought plausible to make claims for absolute correctness of the software in the PPS; instead, it was required to have a *pfd* no worse than 10^{-3}.

The SPS, on the other hand, has much more restricted functionality, and as a consequence is much simpler (and does not depend upon software for its correct operation). Because of this simplicity of design, it might have been possible to claim that the SPS is free of design faults, and thus could be deemed free of failures in operation from such causes.[2] Of course, regulators might have justifiable skepticism when faced with such a claim of "perfection": they might be prepared to accept a high confidence in perfection, given extensive suitable supporting evidence, but not accept *certainty*. It might therefore be plausible to contemplate attaching a probability to the claim of perfection for the SPS and then to combine this in some way with the reliability claimed for the PPS to yield a quantifiable claim for the overall system. This is the key idea in the work reported here; we think it is new.

Asymmetric fault tolerant architectures (i.e., those in which one channel is claimed to be reliable and the other is possibly perfect) are intuitively plausible for other safety-critical systems in addition to nuclear shutdown. For example, airplanes sometimes employ simple "monitors" that check the operation of complex subsystems and trigger

[1] This notion of "on average" efficacy is similar to that used to support claims for other software engineering processes. We know that certain software engineering processes—e.g., different kinds of testing or static analysis—are effective in improving the resulting reliability of the systems to which they are applied [18]. But we usually know very little about *how much* they deliver even on average, and generally nothing at all about what contribution they make to a *particular* system's development (because the particular can be very different from the average). This means that knowledge about the software engineering processes used to build a system is of limited use in estimating its operational reliability.

[2] Of course, the SPS could fail as a result of hardware component failures—as could the PPS. But there are good reasons to believe that such contributions to unreliability are better understood and are small compared to those arising from design faults. For simplicity here and later in this paper we shall ignore hardware failures, but it is easy to extend the modeling to include them.

higher-level fault recovery when they detect problems, while some heart pacemakers and defibrillators have a purely hardware "safety core" that provides backup to a more complex firmware-driven primary system. Whilst these approaches are recognized as attractive ways for *achieving* reliability and safety, it is less widely appreciated that the asymmetry of the architecture can also aid in *assuring* that the system is sufficiently reliable (or safe), if we can exploit the possibility of perfection for the second channel.

In Section 2 of the paper this idea will be examined in some detail. For clarity, the account will be conducted in terms of a 1oo2 demand-based system, such as the Sizewell protection system, but (as seems likely now, even for these nuclear systems) both channels will be assumed to be software-based.

A 1oo2 architecture reduces the likelihood of failures of omission, but may raise those of commission (i.e., uncommanded activations). Section 3 considers these failures of commission, and Section 4 combines the previous results and applies them to "monitored architectures" in which the behavior of a complex operational channel is monitored by a simpler channel that can trigger higher-level fault recovery when safety properties are violated. Architectures of this kind are among those recommended for commercial airplanes [40]. Section 5 presents our conclusions.

2 Reasoning About a One-Out-Of-Two System with a Possibly Perfect Channel

The first part of the reasoning here is similar to that used in the earlier paper [17]. The main novelty is that the present paper establishes *conditional independence* of failure of one channel and imperfection of the other[3] and, in the second part of this section, introduces a proper modeling of the epistemic uncertainty about the parameters of the original work. In giving presentations of this work, we have found that some are surprised by, and others are skeptical of our results; we attempt to allay concern by presenting our arguments in some detail.

We begin by outlining our notion of *failure*; this is described in more detail in Section 2.3 where we examine the estimation of model parameters, but it is useful to narrow the interpretation of the term before we proceed.

Our context is a critical system where the events that constitute a critical failure are defined by regulation (e.g., "inability to continue safe flight and landing" [11]) or as the top-level claims of an assurance or safety case [5, 13]. Through design iterations interleaved with hazard analysis, fault-tree analysis, and other systems engineering processes, we arrive at a level of design that identifies a subsystem to be implemented as a 1oo2 architecture. The process of system decomposition and safety analysis will have identified the critical properties of this subsystem, and will have allocated a reliability requirement to it. Our focus is on this subsystem and in our analysis we take its critical properties (whose violations constitute its failures) and its reliability goal as given, but we assume that these have been chosen appropriately for its rôle in the overall system. This generally means that the critical properties are directly derived from the system safety case, rather than stated in terms of the detailed requirements

[3]Two events X and Y are conditionally independent given some third event Z when $P(X \text{ and } Y \mid Z) = P(X \mid Z) \times P(Y \mid Z)$; we say the independence of X and Y is conditional on Z. Z may be the event that some model parameter takes a known value. Unconditional independence is $P(X \text{ and } Y) = P(X) \times P(Y)$ and is not, in general, implied by conditional independence.

for the subsystem—because the latter are often wrong (we will see an example of this in Section 4).

We can now focus on our object of study, which is a 1oo2 demand-based (sub)system comprising two channels, A and B. Channel A is assumed to have sufficient functionality, and thus complexity, for it to be infeasible to claim it to be fault-free. We thus have to assume that in operation it will eventually fail, and the criterion of its acceptability (when judged alone) will be that it fails infrequently—that is, its probability of failure on a randomly selected demand, pfd_A, is acceptably small. Failure here refers to violation of the critical properties allocated to the 1oo2 subsystem in the system decomposition described earlier. The kinds of evidence that could be used by assessors to gain confidence that pfd_A is small include: seeing lots of operational testing without failure, high quality of developers and development process, no faults found in extensive static analysis, etc.

Channel B, on the other hand, has been kept very simple (its functionality is limited and its implementation straightforward) and thus is open to a claim of "perfection." This word has been chosen deliberately for its convenience, but in the knowledge that it comes with extensive baggage. In this context it means just that this channel of software will not fail however much operational exposure it receives; it is "fault-free" if we assume that failures can and will occur if and only if faults are present. "Perfection" is preferred to "correctness" because the latter is judged with respect to requirements that are often quite detailed, whereas we reserve "perfection" for just the absence of the critical failures. There is a possibility that this channel is *not* perfect, and so we speak of it as "possibly perfect." We use "possibly" perfect rather than other adjectives such as "plausibly" perfect because we will wish to refine our assessment as evidence becomes available: thus we begin from the neutral stance that the channel's perfection is "possible," and may revise that to "plausible," or even "likely," as supporting evidence becomes available. In our formal treatment, we use the neutral term "possibly" perfect and employ probability to express the "degree" of perfection (or, rather, *im*perfection).

In particular, we will henceforth use pnp_B to denote the probability that the channel B is *not* perfect. Evidence allowing assessors to have confidence that the probability pnp_B is small could include: a formal specification and high confidence that this really does accurately capture the critical properties, formal proof that the software implementation is correct with respect to this specification, and absence of failures in test, etc.

We will often speak of A and B as "software" because it is expected that the bulk of their implementation will be realized in this way, but it is important to note that our assessments of reliability and perfection refer to the totality of their design and implementation, and flaws therein, not just the parts that are explicitly software. It is also important that the 1oo2 system is pared down to just the two channels, with a minimum of coordinating mechanism, because we wish to derive the reliability of the system from properties of the channels, and this will be invalid if the subsystem contains a lot of additional mechanism. (We do account for the coordinating mechanism in Section 4.)

There are two kinds of uncertainty involved in reasoning about the reliability of the system. The first is *aleatory uncertainty*, or "uncertainty in the world," the second is *epistemic uncertainty*, or "uncertainty about the world" [28]. Aleatory uncertainty can be thought of as "natural" uncertainty that is *irreducible*. For example, if we were

to toss a coin we would not be able to predict with certainty whether it would fall as "heads" or as "tails." This uncertainty cannot be eliminated, and any predictions about future tosses of the coin can only be expressed as probabilities. This is true even when we know the probability of heads, which we denote p_H, on a single toss of the coin, such as when the coin is "fair" and $p_H = 0.5$. Real coins, of course, may not be fair, so there will be uncertainty about the value of the parameter p_H. This is *epistemic* uncertainty, and it is *reducible*: in the case of the coin, we could toss it very many times and observe the frequency of heads in the sequence of tosses. This frequency is an estimate of p_H and the estimate gets closer to the true value of p_H as the number of observed tosses increases (there are limit theorems in probability that detail the nature of this convergence), so the uncertainty reduces.

In much scientific modeling, the aleatory uncertainty is captured conditionally in a model with unknown parameters, and the epistemic uncertainty centers upon the values of these parameters—as in the simple example of coin tossing.[4] For our 1oo2 system the two probabilities, pfd_A and pnp_B, are parameters that capture two types of aleatory uncertainty. For a randomly selected demand, Channel A either does, or does not, fail; pfd_A is just the chance that it *does* fail. As in the case of a coin toss, this chance can be given a classical frequentist interpretation, as the proportion of failures in a hypothetical infinitely large sequence of independently selected demands.

Similarly, Channel B either is, or is not, perfect; pnp_B is just the chance that it is *not* perfect. It is a little harder to give a frequentist "in the world" interpretation to this probability. One approach, which follows the spirit of [10, 16], is to think of the program for B as a random selection from the set of all programs, with the probability distribution of the selection being determined by the problem to be solved and the character of the software development and assurance processes employed. Each program can be assigned an indicator variable that takes the value 0 if the program is perfect for B's problem and 1 if it is not; pnp_B is then the expected value of the indicator variable over the selection distribution. Just as the nature of the problem and the character of the software development and assurance processes determine the aleatory selection distribution, so information about these will later influence the assessors' judgment about the epistemic uncertainty concerning pnp_B.

A result that we shall prove in Section 2.1 is that these two parameters are sufficient to describe the uncertainty about the *system* probability of failure. Specifically, conditional on knowing the actual values p_A and p_B of pfd_A and pnp_B, respectively, the system probability of failure is no worse than $p_A \times p_B$. The epistemic uncertainty here solely concerns the values of p_A and p_B: assessors will not know these values with certainty. Their beliefs about these values—and here we adopt a Bayesian subjective interpretation of probabilities—will be represented by a joint distribution

$$F(p_A, p_B) = P(pfd_A < p_A, pnp_B < p_B) \tag{1}$$

and their (conservative) probability for system failure will then be

$$\int_{\substack{0 \le p_A \le 1 \\ 0 \le p_B \le 1}} p_A \times p_B \, dF(p_A, p_B). \tag{2}$$

[4]The situation can be more complicated than this, of course. There may be uncertainty about which of several models is "correct," and this uncertainty may not be captured by a parametric model.

Generally, assessors' beliefs about (p_A, p_B) will not be independent: that is, the bivariate distribution (1) does not factorize into a product of the marginal distributions of the random variables pfd_A and pnp_B and so it is not straightforward to derive a numerical value for (2). This is an important issue which is examined in Section 2.2

Our reasoning about the reliability of the system now proceeds in two stages, the first involving the aleatory uncertainty, the second the epistemic uncertainty.

2.1 Aleatory Uncertainty

We shall assume in what follows that our interest centers upon the system reliability expressed as the probability of its failure upon a (operationally) randomly selected demand. Note that other things might be of interest, for instance the probability of surviving a specified number of demands (e.g., the number expected to be encountered during a specific mission, or during the lifetime of the system), but we shall not consider these here.

For aleatory uncertainty we have, by the formula for total probability, the following conditional *pfd*.

$$
\begin{aligned}
&P(\text{system fails on randomly selected demand} \mid pfd_A = p_A, pnp_B = p_B) \qquad (3)\\
&= P(\text{system fails} \mid A \text{ fails},\ B \text{ imperfect},\ pfd_A = p_A,\ pnp_B = p_B)\\
&\quad \times P(A \text{ fails},\ B \text{ imperfect} \mid pfd_A = p_A, pnp_B = p_B)\\
&\quad + P(\text{system fails} \mid A \text{ succeeds},\ B \text{ imperfect}, pfd_A = p_A,\ pnp_B = p_B)\\
&\quad \times P(A \text{ succeeds},\ B \text{ imperfect} \mid pfd_A = p_A,\ pnp_B = p_B)\\
&\quad + P(\text{system fails} \mid A \text{ fails},\ B \text{ perfect}, pfd_A = p_A,\ pnp_B = p_B)\\
&\quad \times P(A \text{ fails},\ B \text{ perfect} \mid pfd_A = p_A,\ pnp_B = p_B)\\
&\quad + P(\text{system fails} \mid A \text{ succeeds},\ B \text{ perfect}, pfd_A = p_A,\ pnp_B = p_B)\\
&\quad \times P(A \text{ succeeds},\ B \text{ perfect} \mid pfd_A = p_A,\ pnp_B = p_B).
\end{aligned}
$$

The last three terms of the right hand side of (3) are zero: the system does not fail if either A succeeds or B is perfect. We shall assume conservatively that if B is not perfect, then it will fail with certainty whenever A fails, as in the original version of this work [17]: B brings no benefit if it is not perfect. This (very) conservative assumption is important in the following reasoning where, in the first term on the right hand side of (3), the first factor in the product is replaced by 1, to yield:

$$
\begin{aligned}
&P(\text{system fails on randomly selected demand} \mid pfd_A = p_A,\ pnp_B = p_B)\\
&\leq P(A \text{ fails},\ B \text{ imperfect} \mid pfd_A = p_A,\ pnp_B = p_B)\\
&= P(A \text{ fails} \mid B \text{ imperfect}, pfd_A = p_A,\ pnp_B = p_B) \qquad (4)\\
&\quad \times P(B \text{ imperfect} \mid pfd_A = p_A,\ pnp_B = p_B).
\end{aligned}
$$

Now, the fact that B is imperfect tells us nothing about whether or not A will fail on *this demand*. So we have

$$
\begin{aligned}
&P(A \text{ fails} \mid B \text{ imperfect}, pfd_A = p_A,\ pnp_B = p_B) \qquad (5)\\
&= P(A \text{ fails} \mid pfd_A = p_A,\ pnp_B = p_B)\\
&= P(A \text{ fails} \mid pfd_A = p_A)
\end{aligned}
$$

and so, substituting (5) in (4),

$$\begin{aligned} & P(\text{system fails on randomly selected demand} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad \leq \ P(A \text{ fails} \mid B \text{ imperfect}, pfd_A = p_A,\ pnp_B = p_B) \\ & \qquad \times P(B \text{ imperfect} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad = \ P(A \text{ fails} \mid pfd_A = p_A) \times P(B \text{ imperfect} \mid pnp_B = p_B) \\ & \quad = \ p_A \times p_B. \end{aligned} \tag{6}$$

This argument follows that in [17] quite closely; the difference is that here we have shown that there is *conditional independence* between the events "A fails" and "B imperfect" when pfd_A and pnp_B are known.

The importance of this independence result can be seen by contrasting the situation here with the more usual 1oo2 system, where both channels are sufficiently complex that neither is open to a believable claim of perfection. In the latter case, we cannot apply the reasoning in step (5) and so it is not possible to claim independence of failures in the two channels, *even conditionally* given pfd_A and pfd_B (in an obvious notation). Instead, even at this stage of reasoning where we are concerned only with aleatory uncertainty, we need to deal with dependence of failures. Thus, it is shown in [16], for example, that the conditional probability of such a system failing on a randomly selected demand is

$$pfd_A \times pfd_B + Cov(\theta_A, \theta_B)$$

where θ_A, θ_B are the difficulty function random variables for the two channels. So assessors' beliefs about pfd_A and pfd_B are not alone sufficient for them to reason about the reliability of the system: they also need to know the covariance of the difficulty functions. And this is before we consider the problem of epistemic uncertainty concerning parameters such as pfd_A, pfd_B, and $Cov(\theta_A, \theta_B)$, which would involve further dependencies.

In contrast, (4–6) above show that knowledge of the values of pfd_A and pnp_B are sufficient for a complete description of aleatory uncertainty via the (conservative) conditional independence expression for the system pfd, (6).

Lest it seem that a particular choice was needed at equation (4), it is worth observing that this could equally have been written as

$$\begin{aligned} & P(\text{system fails on randomly selected demand} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad \leq \ P(A \text{ fails},\ B \text{ imperfect} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad = \ P(B \text{ imperfect} \mid A \text{ fails}, pfd_A = p_A,\ pnp_B = p_B) \\ & \qquad \times P(A \text{ fails} \mid pfd_A = p_A,\ pnp_B = p_B), \end{aligned} \tag{7}$$

where the conditioning is reversed in (7).

Here, the failure of A on this demand tells us nothing about the imperfection of B, since this is a global property with known probability p_B, and so we have

$$\begin{aligned} & P(\text{system fails on randomly selected demand} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad \leq \ P(B \text{ imperfect} \mid A \text{ fails}, pfd_A = p_A,\ pnp_B = p_B) \\ & \qquad \times P(A \text{ fails} \mid pfd_A = p_A,\ pnp_B = p_B) \\ & \quad = \ P(B \text{ imperfect} \mid pnp_B = p_B) \times P(A \text{ fails} \mid pfd_A = p_A) \\ & \quad = \ p_B \times p_A \end{aligned}$$

as before.

The attributes of "perfection" that enable the reasoning employed at (4) and (7) are: first, that its satisfaction guarantees its channel does not fail on a given demand (this is needed to eliminate the last two terms in (3)); second, that it is a global property (i.e., independent of *this* demand, which is needed in (4) and (7)); and third, that it is aleatory (i.e., its uncertainty is irreducible).

Other global properties, such as "has a valid proof of correctness" (whose negation would be "has a flawed proof of correctness"), appear plausible but are not aleatory. Proof of correctness is relative to a specification, and it is possible this specification is flawed. Thus, absence of failure on a given demand (needed for the first point above) is contingent on correctness of the specification, which is an additional parameter and a reducible source of uncertainty.

Hence "perfection" as we have defined it is *precisely* the property required for this derivation. Properties such as "has a valid proof of correctness" are important in assigning an epistemic value to the probability of perfection and are discussed in Section 2.3. First, though, we examine the basic problem of epistemic uncertainty about the model parameters.

2.2 Epistemic Uncertainty and the Structure of $F(p_A, p_B)$

The previous section has derived a conservative bound for the probability of failure on demand for a 1oo2 system, conditional on knowledge of the values of the random variables pfd_A and pnp_B. In practice, of course, pfd_A and pnp_B will not be known with certainty, and this is where the epistemic uncertainty comes in, via the assessors' posterior beliefs represented by the distribution, (1). "Posterior" here means that this distribution takes account of all evidence the assessors have acquired about the unknown parameters.

Practical interest centers upon the *unconditional* probability of system failure on a randomly selected demand. By the formula for total probability, and using the Riemann-Stieltjes integral, this is

$$\begin{aligned} &P(\text{system fails on randomly selected demand}) \\ &= \int_{\substack{0\le p_A\le 1\\ 0\le p_B\le 1}} P(\text{system fails on randomly selected demand} \,|\, pfd_A = p_A,\, pnp_B = p_B)\, dF(p_A, p_B) \end{aligned}$$

so, substituting (6),

$$\le \int_{\substack{0\le p_A\le 1\\ 0\le p_B\le 1}} p_A \times p_B \, dF(p_A, p_B), \tag{8}$$

which provides the formal derivation for (2).

If the assessors' beliefs about pfd_A and pnp_B are independent, so that $F(p_A, p_B)$ factorizes as $F(p_A) \times F(p_B)$, then (8) becomes

$$\begin{aligned} &\int_{0\le p_A\le 1} p_A \, dF(p_A) \times \int_{0\le p_B\le 1} p_B \, dF(p_B) \qquad (9)\\ &= P(A \text{ fails}) \times P(B \text{ imperfect}) \\ &= P_A \times P_B. \end{aligned}$$

That is, the bound on the probability of system failure is just the product of the assessors' posterior probabilities of the events "A fails" and "B imperfect" (i.e., the means of the distributions for pfd_A and pnp_B, respectively, taking account of any experimental and analytical evidence that may be available). In this case the reliability assessment is particularly simple—it involves only the assessment of two parameters.

However, if there is positive association between assessors' beliefs about pfd_A and pnp_B, then (8) will be larger than (9) and reliability assessment cannot be reduced to the simple product of two parameters. Readers may think that this is the situation that will apply in most real-life cases.

The two-stage reasoning—aleatory uncertainty followed by epistemic uncertainty—makes clear the source of any dependence here: it can arise only from the epistemic uncertainty, via the posterior distribution $F(p_A, p_B)$. Even though the events "A fails" and "B is imperfect" are conditionally independent at the aleatory level, assessors' beliefs about the probabilities of these events may not be independent. For example, if assessors see A fail in testing, they may wonder if the developers have understood the requirements correctly, and this may affect their estimate of the probable perfection of B; this does not happen at the aleatory level, where the probabilities of failure and of imperfection are parameters to the model.

Assessors whose beliefs about pfd_A and pnp_B are not independent are faced with the problem of expressing their beliefs as a complete bivariate distribution. Elicitation of experts' subjective probabilities is an active research area in Bayesian statistics, and there have been considerable advances in recent years in techniques and tools: see [29] for a good introduction to this work. We shall not consider this general elicitation problem further here, since for a particular system the elicitation process will depend upon the exact details of the application. Instead, we propose a simplification that will ease this elicitation task in general. Our approach derives from a conservative treatment of common-cause faults in safety cases and is described in the following section.

2.3 Estimation of Model Parameters

One reason—indeed, we will argue, the only sound reason—that assessors' beliefs about the parameters p_A and p_B might not be independent is concern about higher-level faults that could be common to both channels. Common-cause faults are a major issue in the safety assessment of any multi-channel system, so we begin by describing the general context of safety assessment, and the treatment of common-cause faults. We then show how epistemic assessment of the parameters for a 1oo2 system can be performed within this context.

The decision to deploy a safety-critical or other kind of critical system is based on careful assessment of risks, estimating both the likelihood of undesired events, and their potential costs and consequences in terms of environmental harm, loss of life, erosion of national security, loss of commercial confidence or reputation, and so on. The assessment is highly specific to the system concerned and its context, and nowadays is often grounded in an argument-based safety or assurance "case" (see, for example [3, 30, 37, 43, 45]) and similar argumentation implicitly underlies the guidelines (e.g., [31, 32, 40]) developed for standards-based certfication processes. Such a case begins with *claims* that enumerate the undesired loss events and the tolerable failure rate or likelihood associated with them: for example, "as low as reasonably practicable" (ALARP) [44], or "so unlikely that they are not anticipated to occur during the entire

operational life of all airplanes of one type" [11, paragraph 9.e(3)]. *Evidence* is collected about the system, its design, and its processes of development, and then an *argument* is constructed to justify satisfaction of the claims, based on the evidence. This process may recurse through subsystems, with substantiated claims about a subsystem being used as evidence in a parent case.

The upper levels of a safety case will often decompose into subcases based on enumeration of hazards, or failure modes, or architectural components. In the case of multi-channel systems, there will be upper-level analysis of the mechanism (if any) that coordinates them (e.g., a voter or arbiter), and of causes or modes of failure that may be common to all channels. Common causes may include physical faults in shared components such as sensors or in mechanisms that coordinate separate channels, or conceptual faults in the understanding of assumptions or requirements. Much engineering expertise and many associated analytical techniques and tools are available for identifying shared or common mechanisms: for example, common-cause analysis, common-mode analysis, and zonal safety analysis.

By these and other means, arguments and supporting evidence for the correctness of top-level safety requirements and for the absence of common-cause faults in assumptions, mechanisms, and failure modes can be developed in great detail and subjected to substantial analysis and empirical investigation. This is necessary because flaws in these elements could dominate overall system safety and also its probabilistic assessment, no matter how this is performed. We may then propose that assessors' subjective probability for system failure on a randomly selected demand will take the form $C + (1 - C) \times D$, where C is due to residual doubts about the top-level issues of system safety requirements and hazards common to all channels, and D is due to the specific architecture of the system and its constituent channels.

The probability C must be small if the system is to be fit for a critical purpose, but it may not be realistic or useful to strive to make it arbitrarily small. In the UK, for example, it is common for assessors to impose a *claim limit* of around 10^{-4} or 10^{-5} for the probabilistic assessment of any system [42]; in UK civil nuclear systems, there is a specific recommended limit of 10^{-5} for probabilistic assessment of common-cause faults [43, paragraph 172]. These limits can be interpreted as a judgement that assessors will always retain residual doubts about top-level issues and that claims for these, expressed here as C, should be taken only so far, and no further; they ensure that this quantity is sufficiently conservative that it really can absorb the hazards common to all channels and thus allow assessors' beliefs about the probabilities for the separate channels to be treated as independent. In other words, claim limits prevent the value of C being pared away to the point where it threatens this independence.

With this idea in mind, we return to the problem of estimating (8) when assessors' beliefs about pfd_A and pnp_B are not independent. Consider the ways in which dependency might arise. There seem to be only two possibilities: one is concern about the intrinsic difficulty of some demands, the other is common-cause faults. We consider these in turn.

Assessors might observe in testing that the A channel fails more often on some classes of demands than others. But should this affect their beliefs about the probability of perfection of the B channel—that is their beliefs about the existence of demands for which B fails? We argue not, since assurance techniques that can support claims of perfection, such as mechanically checked proofs of correctness, are "uniformly credible"

across all cases: a (mechanically checked) proof is a proof. Failures of A may indeed influence assessment of the failure behavior of B *given imperfection of* B, but that is not the event under consideration (furthermore, our aleatory analysis already makes the conservative assumption that if the B channel is imperfect, then it will fail whenever the A channel does).

Dually, assessors might observe that some parts of the safety case for the possibly perfect channel B are harder (i.e., require more work) than others. But should this affect their confidence in corresponding parts of the reliable channel A and their beliefs about its probability of failure? Again, we argue not: it is accepted that the reliable channel fails with some probability, and we have evidence about that probability (e.g., by statistically valid random testing).

Thus, unlike the case of two reliable channels, we do not think that intrinsic difficulty of some demands should be a cause for dependency in beliefs about pfd_A and pnp_B: failure of the A channel and imperfection of the B channel are such different events, and the assurance methods that provide evidence for estimation of their probabilities are of such different kinds, that dependency should not arise from concern about difficult demands.

Rather, dependency in beliefs must have its root cause in some higher level that is common to both channels, such as shared mechanism, or the interpretation of requirements. One way forward, therefore, building on the treatment described earlier for common-cause faults, is for the assessors to place some probability mass, say C, at the point (1, 1) in the (p_A, p_B)-plane to represent their subjective probability that there are such common faults. The effect of this is that the assessors believe that if there *are* such faults, then A will fail with certainty ($p_A = 1$), B will be imperfect with certainty ($p_B = 1$), and hence $p_A \times p_B = 1$. Hence, there is a probability C that the system is certain to fail on a randomly selected demand (since we assume that if B is imperfect, it always fails when A does). This is clearly conservative, and may be very much so.

The assessors then have the following conservative unconditional probability of failure of the system on a randomly selected demand (again by a version of the formula for total probability).

$$
\begin{aligned}
&P(\text{system fails on randomly selected demand}) \\
&\leq \int_{\substack{0\leq p_A\leq 1\\ 0\leq p_B\leq 1}} p_A \times p_B \, dF(p_A, p_B) \\
&= C \times \int p_A \times p_B \, dF(p_A, p_B \,|\, p_A = p_B = 1) \\
&\quad + (1 - C) \times \int_{\substack{0\leq p_A< 1\\ 0\leq p_B< 1}} p_A \times p_B \, dF(p_A, p_B \,|\, p_A \neq 1, p_B \neq 1) \\
&= C + (1 - C) \times \int_{\substack{0\leq p_A< 1\\ 0\leq p_B< 1}} p_A \times p_B \, dF(p_A, p_B). \qquad (10)
\end{aligned}
$$

If the assessors believe that all dependence between their beliefs about the model parameters has been captured conservatively in C, the conditional distribution in the

second term of (10) now factorizes, and we have:

$$\begin{aligned} & P(\text{system fails on randomly selected demand}) \\ \leq\ & C + (1-C) \times \int_{0 \leq p_A < 1} p_A \, dF(p_A) \times \int_{0 \leq p_B < 1} p_B \, dF(p_B) \\ =\ & C + (1-C) \times P_A^* \times P_B^* \end{aligned} \tag{11}$$

where P_A^* and P_B^* are the means of the posterior distributions representing the assessors' beliefs about the two parameters, each conditional on the parameter not being equal to 1 (i.e., A not certain to fail, B not certain to be imperfect).

If C is small (as it must be if this approach is to be useful), we can replace the factor $(1-C)$ in (11) by 1, and we can substitute P_A, the unconditional probability of failure of the A channel, for P_A^*, and P_B, the unconditional probability of imperfection of the B channel, for P_B^*, to yield the *conservative* approximation

$$C + P_A \times P_B. \tag{12}$$

Assessors may find it easier to formulate their beliefs about these unconditional probabilities. The same simplification can be applied to similar formulas developed in later sections and the resulting approximations will all be conservative.

We now consider and illustrate numeric values that might be assigned to the parameters that appear in Formula (11) or its simplification (12). We use the word "estimation" for the process by which assessors expresses their "expert beliefs" in terms of these values, but we do not mean to imply that the numbers have aleatoric, *in*-the-world meaning, of course. Rather, they are representations of the assessors' beliefs *about* the world.

Bearing in mind the discussion about claim limits, and the facts that a 1oo2 system has little or no mechanism shared between its two channels and that the common top-level requirements are not engineering details but the critical safety requirements, we suggest that 10^{-5} is representative of values that could be plausible estimates for C, the probability of common-cause failure. It then remains to consider the processes for obtaining, or eliciting, P_A^* and P_B^*, or their unconditional variants P_A and P_B.

The first of these is the assessors' probability of failure on demand for the reliable channel. For values down to about 10^{-4}, it is feasible to provide strong and credible evidence by statistically valid random testing [7, 22]. By statistically valid here we mean that the test case selection probabilities are exactly the same as those that are encountered in real operation, and that the oracle (used to determine whether a demand has been executed correctly or not) is sound. See [25] for an example of such testing used to assess the reliability of a reactor protection system. In the event that there is doubt about the representativeness of the test case selection, or of the correctness of the oracle, then this epistemic uncertainty must be incorporated into the assessors' probability P_A^*. Expert assessors might take account of other kinds of evidence as well as that obtained from testing in arriving at their P_A^*. We shall not discuss this issue in detail here, but note that Bayesian Belief Nets (BBNs) are one example of a formalism for handling such disparate sources of evidence and uncertainty (see, for example, [23]).

For the possibly perfect channel, the assessors must express their subjective probability of its imperfection: that is, their degree of belief that there is at least one demand on

which it fails. A rational process for developing this probability uses Bayesian methods to modify a prior belief through the acquisition of evidence to yield a refined posterior belief (see [22, sidebar] for a brief description of this process). The strongest evidence will come from testing and scrutiny and analysis of the design and implementation of the channel concerned. As the claim is "perfection," the analysis will likely include extensive "Verification and Validation" (V&V) of the kinds recommended in guidelines such as DO-178B [31] for airplanes, or MISRA [26] for cars. More forward-looking guidelines such as the forthcoming DO-178C update to DO-178B may also countenance formal verification. Notice that if testing reveals a failure, or if formal verification or other analysis reveals a fault, then the claim of perfection is destroyed and the channel must be replaced in its entirety.[5] In the following, therefore, we consider only the case in which testing and verification reveal no failures.

Formal verification and other analyses can be performed to various degrees of thoroughness: they may examine just the abstract design of the software and its algorithms, or detailed models developed in a model-based design framework, or the executable code, or all of these. In addition, the formalizations and analyses may extend to the non-software elements that are part of the channel—for example, its sensors and actuators, the physical processes that they monitor and control, and its environment—possibly including human operators. The properties verified may range from simple absence of runtime anomalies, as may be guaranteed by static analysis, to full functional correctness with respect to a detailed requirements specification. The subjective probability of imperfection will obviously be influenced by consideration of how comprehensive is the analysis, and the extent of what has *not* been formally verified or otherwise examined in comparable detail; it will also be tempered by consideration of the complexity of the design and its implementation, and possibly by residual concerns about the soundness of the formalization, the analyses, and the tools employed. The significance and relative weights of these concerns in assessing a probability of imperfection are considered in [36], where is it suggested that values for P_B^* in the range 10^{-2}–10^{-4} are credible.

In summary, we believe that it is plausible and conservative that well-designed and highly assured 1oo2 systems can support claims such as 10^{-5} for C, the probability of failure due to causes common to both channels, 10^{-3} for P_A^*, the probability of failure by the reliable channel, and 10^{-2} for P_B^*, the probability of imperfection of the possibly perfect channel. Substituting these values in (11) we obtain $10^{-5} + (1 - 10^{-5}) \times 10^{-3} \times 10^{-2}$ and thereby substantiate a conservative assessment of about 2×10^{-5} for the probability of failure on demand of the overall system. This assessment and its constituent quantities are comparable to those that have been employed for real systems, but we believe this is the first account that provides an argument by which they can be rigorously defended.

Because the idea is new, there is no experience in assessing probabilities of imperfection, so we really have little idea what values might be assessed in practice and therefore deliberately used a very conservative value in this example. However, as we outline in [36], and intend to argue more fully in future [21], the traditional processes of software assurance performed in support of DO-178B [31] (the guidelines for certification of aircraft software) and similar certification regimes are best understood as

[5] We are speaking here of the testing and verification performed for assurance and evaluation; discovery and repair of failures and faults in earlier stages of development may be acceptable in some lifecycle processes.

developing evidence of possible perfection. The probability of failure for such highly assured software is then bounded by the product of its probability of imperfection and its conditional probability of failure, if imperfect. Software developed and evaluated to the higher "Design Assurance Levels" of DO-178B can be used in contexts requiring very low probabilities or rates of failure (e.g., 10^{-9} per hour), and historical experience validates this judgment. Littlewood and Povyakalo [20] show that when faced with increasing evidence for failure-free operation, our beliefs should tend toward attributing this to the software being perfect rather than having a very small conditional probability of failure. Thus, we believe that precedent and experience are able to support credible assessments of very low probabilities of imperfection for suitable software, and we are hopeful that consensus will develop in support of this position.

The relative sizes of the quantities appearing in the example above are worth some comment. It is plausible that assessors could assign a value as small as 10^{-4} to P_A^* and, as argued above, a similar value seems feasible for P_B^*; their product is then 10^{-8}, which approaches the range required for ultra-high dependability in mass transit systems such as trains. The fly in the ointment is common-cause failure, where it may be difficult to assess a probability smaller than 10^{-5} to C, and this therefore sets the overall limit on claimed reliability. There are two ways to interpret this observation; one is to conclude that our analysis is cautionary and that requirements and claims for ultra-high dependability should be viewed skeptically, especially if they depend on assigning very low probabilities to common-cause events. We note that the Airbus A340 fuel emergency to be discussed in Section 4 was essentially a common-cause failure.

The other interpretation is that our approach to simplifying the problem of epistemic estimation is inadequate for applications requiring ultra-high dependability: it conflates genuine common-cause failure (for whose likelihood we may truly be able to assess a very low probability) with lack of independence in beliefs about pfd_A and pnp_B. This is an area where further research and investigation of different approaches seem warranted. One alternative approach has been developed by Littlewood and Povyakalo [19], following circulation of drafts of the present paper. This approach asks assessors to state beliefs about the *marginal* distributions of the two parameters p_A and p_B, and then calculates conservative approximations to the joint distribution (8) that do not assume independence. One of their most attractive results is that if we have a *confidence bound* $1 - \alpha_A$ on an assessment P_A for pfd_A (i.e., $P(pfd_A < P_A) > 1 - \alpha_A$), and $1 - \alpha_B$ on an assessment P_B for pnp_B, then pfd_S, the probability of failure on demand for the system, satisfies

$$P(pfd_S < P_A \times P_B) > 1 - (\alpha_A + \alpha_B).$$

In words, we "multiply the claims and add the doubts" [19]. For example, if we are 95% confident that $pfd_A < 10^{-4}$ and 90% confident that $pnp_B < 10^{-4}$, then we can be 85% confident that the system probability of failure on demand is less than 10^{-8}. Whilst this approach delivers only confidence bounds and does not support calculation of the actual posterior probability of failure on demand of the system, (8), it may nevertheless be possible to use these confidence bounds within a wider safety case.

We return to this topic in Section 4, where we develop an analysis for the kind of monitored architectures proposed for aircraft, but first we need to examine failures of commission.

3 Type 2 Failures, or Failures of Commission

In this section, we turn from failures of omission to those of commission: those that perform an action (such as shutting down a reactor) when it is not required. In some system contexts, this kind of failure can have serious safety consequences, while in others it may be a costly nuisance, and in yet others merely an annoying "false alarm." We study failures of this kind because they are of some importance in themselves, and because they lay the groundwork for our analysis of monitored architectures in Section 4.

In a 1oo2 system, either channel can put the system into a safe state. Hence, the failure of one channel to bring the system to a safe state when this should be done (i.e., on a "demand") can be compensated by the other channel working perfectly. But, dually, the 1oo2 arrangement means that a single channel can force the system to a safe state when it is not necessary to do so (i.e., when there is no demand or, as we shall sometimes say, on a "nondemand"). We can call the failure to bring the system to a safe state when it is necessary to do so a "Type 1" failure (a failure of omission), and the dual failure that does this unnecessarily a "Type 2" failure (a failure of commission).

Although we refer to Type 1 and Type 2 failures as duals, their treatment requires careful consideration because the demands and nondemands that underlie them are somewhat different in character. Intuitively, a demand is an event at some *point* in time, whereas a nondemand is the absence of a demand over a *period* of time. We unify the two by adjusting our treatment to consider the *rate* at which demands occur.

The channels of the systems we are interested in typically operate as cyclic processes executing at some fixed rate, such as so many times a second; on each execution, they sample sensors, perform some computation, possibly update their internal state, and decide whether to act (e.g., to shut down the reactor) or not. As time passes, there will be very many executions, only a few of which might be demands—i.e., represent states in which the channels are required to act. We suppose that whether any particular execution is a demand or not is independent of any other execution, so the numbers of demands in disjoint time intervals are independent of each other (i.e., "independent increments" in the terminology of statistics) and the probability distribution of the number of demands in a time interval depends only on the length of the interval (i.e., "stationary increments"). Under these conditions, the occurrence of demands will be approximately a Poisson process [33] with rate d, say. For the applications we have in mind, there will be very great disparity between the durations of the cycles and the expected times between successive demands, so this approximation will be very close. In what follows, therefore, we shall use the continuous time formalism of the Poisson process. So

$$d = \lim_{\delta t \to 0} \frac{P(1 \text{ demand in } (t,\, t + \delta t))}{\delta t}. \tag{13}$$

We define failure rates in a similar way. Then, for Type 1 failures we have

$$\begin{aligned}
&\text{1oo2 system Type 1 failure rate} \\
&= \lim_{\delta t \to 0} \frac{P(\text{1 demand and system fails in } (t,\, t+\delta t))}{\delta t} \\
&= \lim_{\delta t \to 0} \frac{P(\text{system fails} \,|\, \text{1 demand in } (t,\, t+\delta t)) \times P(\text{1 demand in } (t,\, t+\delta t))}{\delta t} \\
&= \lim_{\delta t \to 0} P(\text{system fails} \,|\, \text{1 demand in } (t,\, t+\delta t)) \times \frac{P(\text{1 demand in } (t,\, t+\delta t))}{\delta t} \\
&\leq \left(C + (1-C) \times P_A^* \times P_B^*\right) \times d
\end{aligned}$$

where the first term comes from the epistemic formula (11) and the second from (13). Using (12) and rearranging terms we also have the conservative simplification

$$\text{1oo2 system Type 1 failure rate} \leq d \times (C + P_A \times P_B). \tag{14}$$

For Type 2 failures due to the A channel, we have

$$\begin{aligned}
&\text{1oo2 system Type 2 failure rate due to } A \\
&= \lim_{\delta t \to 0} \frac{P(\text{no demand and } A \text{ activates in } (t,\, t+\delta t))}{\delta t} \\
&= \lim_{\delta t \to 0} \frac{P(A \text{ activates} \,|\, \text{no demand in } (t,\, t+\delta t)) \times P(\text{no demand in } (t,\, t+\delta t))}{\delta t} \\
&= \lim_{\delta t \to 0} \frac{P(A \text{ activates} \,|\, \text{no demand in } (t,\, t+\delta t))}{\delta t} \times P(\text{no demand in } (t,\, t+\delta t)).
\end{aligned}$$

The second term here is conservatively approximated by 1. Then, taking limits, the first term is the failure rate of Channel A with respect to Type 2 failures, whose epistemic value (i.e., mean of the assessors' posterior distribution) we will denote by F_{A2}. Thus, we have

$$\text{1oo2 system Type 2 failure rate due to } A \leq F_{A2}. \tag{15}$$

We could perform an exactly similar calculation for Type 2 failures due to the B channel, but this would not exploit the possible perfection of that channel. Accordingly, we suppose that the possible perfection of B extends to Type 2 failures and that p_{B2} is its aleatory probability of imperfection with respect to those failures. We then have

$$\begin{aligned}
&P(B \text{ activates} \,|\, \text{no demand in } (t,\, t+\delta t)) \\
&\quad = P(B \text{ activates} \,|\, B \text{ imperfect, no demand in } (t,\, t+\delta t)) \times P(B \text{ imperfect}) \\
&\qquad + P(B \text{ activates} \,|\, B \text{ perfect, no demand in } (t,\, t+\delta t)) \times P(B \text{ perfect}).
\end{aligned}$$

The second term in this sum is zero (B does not activate on a nondemand if it is perfect with respect to Type 2 failures), so the right hand side reduces to

$$P(B \text{ activates} \,|\, B \text{ imperfect, no demand in } (t,\, t+\delta t)) \times p_{B2}.$$

The calculation given earlier for the A channel can now be reproduced for the B channel, but working at the aleatory level and substituting the expression above in the final line. This gives

$$\begin{aligned}
&\text{1oo2 system Type 2 aleatory failure rate due to } B \\
&\leq \lim_{\delta t \to 0} \frac{P(B \text{ activates} \,|\, B \text{ imperfect, no demand in } (t,\, t+\delta t))}{\delta t} \times P(\text{no demand in } (t,\, t+\delta t)) \times p_{B2}.
\end{aligned}$$

Here, the second term is conservatively approximated by 1. Then, taking limits, the first term is the aleatory failure rate of B with respect to Type 2 failures, *conditional* on B being imperfect with respect to these failures, which we will denote by $f_{B2|np}$.

$$\text{1oo2 system Type 2 aleatory failure rate due to } B \leq f_{B2|np} \times p_{B2}. \tag{16}$$

The right hand side is the product of two parameters describing aleatory uncertainty. We now have to consider assessors' epistemic uncertainty about these parameters to derive a posterior expression for the failure rate. A fully accurate treatment requires integrating with respect to a joint probability function $G(f_{B2|np}, p_{B2})$ just as we did with the function $F(p_A, p_B)$ in Section 2.2. However, it may be that assessors' beliefs about these parameters will be independent (why should the rate of failure, if imperfect, be dependent on the probability of imperfection?) so that the distribution G can be factorized and the failure rate due to Type 2 errors in B is then given by

$$\text{1oo2 system Type 2 failure rate due to } B \leq F_{B2|np} \times P_{B2} \tag{17}$$

where $F_{B2|np}$ is the assessors' posterior failure rate of B with respect to Type 2 failures, assuming B is imperfect with respect to these failures, and P_{B2} is the posterior probability of B being imperfect with respect to Type 2 failures. Each of these is the mean of the posterior distribution of its random variable considered alone.

We can now assess the overall merit of a 1oo2 system in terms of its *risk* per unit time, taking account of both Type 1 and Type 2 failures, as follows. Risk is the product of the rate or probability of a failure and the severity of its worst-case consequences. Denote the worst-case severity of a Type 1 failure by c_1, and those of Type 2 failures by the A and B channels by c_{A2} and c_{B2}, respectively. (Since the A and B channels may bring the system to a safe state in different ways—e.g., by dropping the control rods or by poisoning the moderator, respectively—it is reasonable to assess different severities to their Type 2 failures.) In some applications, a Type 2 failure may have no safety consequences, so c_{A2} and c_{B2} will be assessed as zero. But even in these cases there are likely to be economic or other losses (e.g., reputation) associated with a Type 2 failure and these can be included in a broader calculation of risk, resulting in nonzero assessments for c_{A2} and c_{B2}. We will refer to c_1, c_{A2} and c_{B2} as *costs*, presuming that worst-case safety consequences and economic and other losses are normalized into some financial scale.

The rates for the three kinds of system failure are given by (14), (15), and (17), respectively. Adding the products of these and their costs yields the total risk:

$$\text{1oo2 } \textit{risk} \text{ per unit time} \leq c_1 \times d \times (C + P_{A1} \times P_{B1}) + c_{A2} \times F_{A2} + c_{B2} \times F_{B2|np} \times P_{B2}. \tag{18}$$

We may assume that the cost c_1 of a Type 1 failure is large, and hence that extensive design and engineering effort is focused on ensuring that both the demand rate d and the failure rate $(C + P_{A1} \times P_{B1})$ are small, so that the contribution to overall risk by the first term in this sum is acceptable.

The second and third terms in the sum (18) are not reduced by a demand rate so, when c_{A2} and c_{B2} are nonzero, their contribution to overall risk will be small only if all their other components are small. The risk from a Type 2 failure of the main A channel is inherent in any safety system, so we may assume that the failure rate F_{A2} is acceptable, relative to the cost of this type of failure. The risk due to a Type 2 failure

of the B channel, however, is an additional factor introduced by the decision to employ a 1oo2 architecture. Hence, we will need to be sure that the contribution from the product $c_{B2} \times F_{B2|np} \times P_{B2}$ is an acceptable price to pay for the reduction in risk that the architecture brings to Type 1 failures.

In the above analysis we have assumed that d, c_1, c_{A2} and c_{B2} are known. If they are not, then they need to be treated as a part of a more general epistemic analysis. This is simple if it can be assumed, as may be reasonable in some cases, that assessors' beliefs about these parameters are independent of their beliefs about the parameters representing different probabilities and rates of failure and imperfection. We shall not pursue this further here.

4 Monitored Architectures

In this section we examine a different class of architectures from the 1oo2 case considered so far. These architectures employ an *operational* channel that is completely responsible for the functions of the system concerned, and a *monitor* channel that can signal an "alarm" if it deems the operational channel to have failed or to be causing unsafe behavior. Other systems will ignore the outputs of a system whose monitor is signaling an alarm, so the monitor transforms potential "malfunction" or "unintended function" into "loss of function" (another interpretation is that the purpose of the monitor is to avoid transitions into hazardous states). Loss of function may still be a dangerous condition, so there must generally be some larger system that responds to the alarm signal and compensates in some way for the lost function. This is really, therefore, an architecture for *sub*systems; we refer to it as a *monitored architecture*.

Monitored architectures are attractive for subsystems of highly redundant systems where occasional loss of function is anticipated and it is likely that some other subsystem can take over the functions of the one whose monitor has signaled the alarm, and the latter subsystem can be shut down or otherwise isolated.

Aircraft systems have this characteristic, as there is a requirement that no single failure may cause a "catastrophic failure condition" [11, paragraph 5]; the redundancy (e.g., multiple engines, multiple sources of air data, and internal redundancy in systems such as autopilot, flight management, autoland, autothrottle, FADEC, etc.) is primarily present to cope with physical failures (of a mechanical element or of a computer), but there is increasing interest in using it to handle software faults. The software in flight-critical functions is supposed to be almost certainly fault-free and guidelines such as ARP4754 [40], DO-178B [31], and DO-297 [32] are intended to ensure that it is so. However, the reliability requirements for flight critical functions (those that could cause catastrophic failure conditions) are so high (e.g., $1 - 10^{-9}$ per hour, sustained for the duration of the flight [11, paragraph 10.b])[6] that there is some concern that no amount of development assurance can deliver sufficient confidence that its embedded software is free of faults.

A position paper by a team of experts representing certification authorities of North and South America, and Europe [8] expresses this concern, and suggests that "fault-

[6] An explanation for this figure can be derived [24, page 37] by considering a fleet of 100 aircraft, each flying 3,000 hours per year over a lifetime of 33 years (thereby accumulating about 10^7 flight-hours). If hazard analysis reveals ten potentially catastrophic failure conditions in each of ten systems, then the "budget" for each is about 10^{-9} if such a condition is not expected to occur in the lifetime of the fleet.

tolerance techniques should be applied (such as, diverse and redundant implementations or simple design)..."[7] Monitored architectures are an approach for achieving this and our proposal is that the monitor channel should be supported by a credible claim of possible perfection. One way to do this would be to develop the monitor channel as a very simple piece of software and to formally verify it against its requirements, in much the same way as we suggested for the possibly perfect channel in the 1oo2 architecture.

Alternatively, rather than formally analyzing the monitor channel, we could formally *synthesize* it from its requirements. There is a topic in formal methods known as *Runtime Verification*, whose technology provides methods for automated synthesis of monitors for formally specified properties [12]. Given requirements specified in a language such as EAGLE or RULER [4], the methods of runtime verification can automatically generate an efficient (and provably correct) monitor that will raise an alarm as soon as a requirement is violated.

We refer to monitors developed using either formal analysis or synthesis as *formal monitors*, and now turn to consideration of the requirements that they monitor.

One obvious source of requirements is the documentation for the operational channel being monitored. The problem with this choice is that even the "high-level software requirements" of guidelines such as DO-178B are focused on software functions rather than (sub)system safety properties, and operational software is often robustly correct with respect to these requirements (due to the effectiveness of practices such as those recommended by DO-178B). That is to say, the overall (sub)system may have failed or be operating in an unsafe manner, but each individual software component is operating correctly according to its requirements.

A recent in-flight incident illustrates this topic. It concerns an Airbus A340-642, registration G-VATL, which suffered a fuel emergency on 8 February 2005 [41]. The plane was over Europe on a flight from Hong Kong to London when two engines flamed out. The crew found that the fuel tanks supplying those engines were empty and those for the other two engines were very low. They declared an emergency and landed at Amsterdam. The subsequent investigation reported that two Fuel Control Monitoring Computers (FCMCs) are responsible for pumping fuel between the tanks on this type of airplane. The two FCMCs cross-compare and the "healthiest" one drives the outputs to the data bus. In this case, both FCMCs had known faults (but complied with the minimum capabilities required for flight); unfortunately, one of the faults in the one judged healthiest was inability to drive the data bus. Thus, although it gave correct commands to the fuel pumps (there was plenty of fuel distributed in other tanks), these were never received.

Another component of the A340 fuel system, the Fuel Data Concentrator (FDC), independently monitors fuel levels and generates a warning when these fall below a threshold. However, this alarm is disregarded by the Flight Warning Computer (FWC) unless both FCMCs have failed. The reason for this is not explained, but a plausible explanation is that it is to avoid Type 2 failures: a correctly functioning FCMC has access to more sensor data and can more accurately judge fuel levels than can the FDC. One of the recommendations of the Incident Report [41, Safety Recommendation 2005-37] is to change the FWC logic so that the FDC can always trigger a low fuel

[7] Observe that the concern underlying this suggeston can be interpreted as a "claim limit" (recall Section 2.3), though of a different kind from that used in nuclear systems: the expert team is saying they have residual doubts about the reliability of any complex single-channel system.

level warning. Observe that the fault that caused the FCMCs to malfunction and that disabled the FDC warning was a conceptual flaw in how the "healthiness" of FCMCs should be assessed, and this gave rise to common-cause faults in the requirements for the FCMCs and FWC.

This example illustrates our claim that there is unlikely to be much benefit in monitoring requirements at or below the component level: not only is critical software generally correct with respect to this level of specification, but faults are often located in these requirements. Furthermore, larger problems may not be manifested at this level (i.e., they are *emergent* above the component level). Instead, we need to monitor properties that more directly relate to the safe functioning of the overall system, and that are more likely to be violated when problems are present.

The claims and assumptions of an argument-based safety case can provide exactly these properties. A safety case for the A340 fuel system would surely include statements about the acceptable distribution of fuel among the different tanks, in addition to minimum levels in the tanks feeding the engines. Properties concerning fuel distribution are different in kind (i.e., "diverse") from those that appear in software requirements, which typically focus on tasks to be performed rather than properties to be maintained (i.e. invariants), and are good candidates for monitoring: unlike low fuel level (the property monitored by the A340 FDC) incorrect fuel distribution could trigger an alarm quite early in the flight and seems unlikely to trigger false alarms.

A monitored architecture therefore has a highly reliable operational channel, whose behavior is derived from its software requirements specification, and a simple formal monitor that has a high probability of perfection with respect to properties taken directly from the system safety case.[8] Analysis of the reliability of such architectures can build on our previous analysis of 1oo2 architectures, with the operational channel taking the part of A and the monitor the part of B, but with some changes that reflect differences in the architectures.

One significant change is that there is no notion of "demand" for the operational channel in the monitored architecture; instead, it continuously performs its assigned operational role, but may occasionally suffer failures that violate a safety requirement. Thus, a monitored architecture has a Type 1 failure if its operational channel violates a safety requirement and its monitor fails to detect that fact. The operational channel will have some failure rate, denoted fr_A, the monitor will have some probability of imperfection with respect to Type 1 failures (i.e., failures of omission), denoted pnp_{B1}, and we seek a rate for Type 1 system failure based on these parameters. The other significant change is that there is no notion of Type 2 failure (i.e., of commission) for the operational channel: only the monitor channel can raise an alarm, and thereby cause a Type 2 system failure by doing so unnecessarily. We suppose that the monitor will have some probability of imperfection with respect to Type 2 failures, denoted pnp_{B2} and we seek a rate for Type 2 system failure based on this parameter.

[8] Of course, the safety case should provide evidence and argument that the operational channel does satisfy those properties. Thus, the monitor is redundant if the safety case is sound; its justification is diversity for protection against the possibility that the safety case is flawed.

We begin with the aleatory analysis for Type 1 system failure; we assume that fr_A has known value f_A and that pnp_{B1} has known value p_{B1}. Then

$$\begin{aligned}
&P(\text{monitored architecture has Type 1 failure in } (t,\, t+\delta t) \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t), B \text{ does not detect failure} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad \leq P(A \text{ fails in } (t,\, t+\delta t), B \text{ imperfect} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \qquad (19) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t) \,|\, B \text{ imperfect}, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\qquad \times P(B \text{ imperfect} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t) \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \qquad (20) \\
&\qquad \times P(B \text{ imperfect} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t) \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \times p_{B1}. \qquad (21)
\end{aligned}$$

Line (20) follows from the one above it because the fact that B is imperfect tells us nothing about the failure of A in this interval. Line (19) follows from the one above it by the following reasoning:

$$\begin{aligned}
&P(A \text{ fails in } (t,\, t+\delta t), B \text{ imperfect} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t), B \text{ imperfect and does not detect failure} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\qquad + P(A \text{ fails in } (t,\, t+\delta t), B \text{ imperfect and detects failure} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad \geq P(A \text{ fails in } (t,\, t+\delta t), B \text{ imperfect and does not detect failure} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}) \\
&\quad = P(A \text{ fails in } (t,\, t+\delta t), B \text{ does not detect failure} \,|\, fr_A = f_A,\, pnp_{B1} = p_{B1}).
\end{aligned}$$

The last line above follows from its predecessor because B must be imperfect if it does not detect failure (of A).

From (21), dividing by δt and taking limits, we obtain

$$\text{Monitored architecture Type 1 aleatory failure rate} \leq f_A \times p_{B1}. \qquad (22)$$

The Type 2 failure rate for the monitored architecture is just that for the monitor channel. We can directly apply the analysis from the 1oo2 case so that (16) becomes

$$\text{Monitored architecture Type 2 aleatory failure rate} \leq f_{B2|np} \times p_{B2} \qquad (23)$$

where $f_{B2|np}$ is the Type 2 failure rate for the monitor channel, conditional on its imperfection with respect to that class of failures.

As with the cases examined earlier, we now need to consider epistemic uncertainty in the two formulas above. For the Type 1 failure rate (22), we will need to consider a joint distribution function $H(f_A,\, p_{B1})$ representing assessors' beliefs about these variables. The first of these is the failure rate of the operational channel, while the second is the probability of imperfection of the monitor. We argue that these are different measures about very different channels: one a fairly complex system designed to meet its requirements (which will be highly operational and removed from the claims of the safety case—though intended to ensure them), the other a very simple and direct check of properties from the safety case. Thus, it is very plausible that assessors' beliefs about these channels will be independent, and so the function $H(f_A,\, p_{B1})$ can be factorized into distributions for the individual variables. However, although the channels are diverse in purpose and design, it is quite likely they will share some implementation mechanisms: for example, a monitor for a fuel system, such as that in the A340 example,

might rely on the same sensors as those used by the operational channel. These shared mechanisms induce potential common-cause failures that must be taken into account.

One approach would be to factor the system explicitly into three components—shared mechanism (e.g., sensors), operational channel, and monitor—and to calculate probabilities over this more complex model. However, we propose a simpler treatment that is similar to that employed to yield (12) in Section 2.2. There, we introduced an element C to absorb failures due to faults in requirements and interpretation that are common to both channels; here, we introduce an element M_1 to absorb failures due to mechanisms that are common to both channels. Thus,

$$\text{Monitored architecture Type 1 failure rate} \leq M_1 + F_A \times P_{B1}, \tag{24}$$

where M_1 is the assessed Type 1 failure rate due to mechanisms shared between the operational and monitor channels, F_A is the posterior failure rate of the operational channel (excluding items in M_1), and P_{B1} is the posterior probability of imperfection (with respect to Type 1 failures) of the monitor channel (again, excluding items in M_1).

The Type 2 failure rate (23) was derived directly from the B channel case in a 1oo2 architecture (16). We argue that the epistemic valuation can similarly be derived from (17) in the 1oo2 case to yield

$$\text{Monitored architecture Type 2 failure rate} \leq M_2 + F_{B2|np} \times P_{B2}, \tag{25}$$

where M_2 is the assessed Type 2 failure rate due to mechanisms shared between the operational and monitor channels, $F_{B2|np}$ is the posterior Type 2 failure rate for the monitor channel, assuming it is imperfect with respect to that class of failures, and P_{B2} is the posterior probability of imperfection of the monitor channel with respect to Type 2 failures. (Although Type 2 failures are due to the monitor channel alone, we must still include the contributions of the mechanisms that it shares with the operational channel.)

Putting these together, we obtain the *risk* of a monitored architecture:

$$\text{Monitored architecture } \textit{risk} \text{ per unit time} \leq c_1 \times (M_1 + F_A \times P_{B1}) + c_2 \times (M_2 + F_{B2|np} \times P_{B2}), \tag{26}$$

where c_1 is the cost of a Type 1 failure and c_2 the cost of a Type 2 failure.

Assuming, as seems likely, that M_1 is much smaller than F_A (since the failures considered in M_1 are included among those in F_A), a relatively modest claim for perfection of the monitor with respect to Type 1 failures (e.g., $P_{B1} = 10^{-3}$) provides significant reduction in the overall risk due to Type 1 failures.

Employing a monitor brings with it the new risk of a Type 2 failure, and responsibility for this is borne by the monitor alone. For this new risk to be acceptable, assuming M_2 is small, we need either a strong claim for perfection of the monitor with respect to Type 2 failures (e.g., $P_{B2} = 10^{-6}$), or the consequence of a Type 2 failure must be modest. Relief from this demanding requirement on P_{B2} is possible if assessors are able to assign a small value to $F_{B2|np}$, the failure rate of the monitor channel, assuming it is imperfect. Statistically valid testing of the monitor channel provides a basis for assessing $F_{B2|np}$ and could yield values on the order of 10^{-3}, thereby potentially reducing the requirement on P_{B2} to a similar value. Obviously, the tests must reveal no failures, since then the channel would definitely be imperfect.

Whatever the contribution of $F_{B2|np}$, there should be an inverse relationship between the product $F_{B2|np} \times P_{B2}$ and the cost c_2. For the case of the A340 fuel management system described earlier, it is likely that the response to an alarm raised by a monitor would be a warning to the pilots, who would then follow the standard troubleshooting and emergency procedures established for possible faults in the fuel system. Thus, the consequences of a false alarm are modest and commensurate with a correspondingly modest claim for perfection of the monitor with respect to Type 2 failures.

On the other hand, monitoring mechanisms on American Airlines Flight 903 of 12 May 1997 (an Airbus A300) inappropriately raised an alarm indicating failure of one of the avionics buses. This caused an automated bus reset, which in turn caused the Electronic Flight Instrumentation System (EFIS), which is responsible for displaying instruments on the cockpit monitors, to go blank for a while. At that time, the pilots were attempting to recover from a major upset (the plane was in a succession of stalls) and the loss of all primary instruments at this critical time ("when the situation was at its gravest") jeopardized the recovery [27]. Here, the cost of a Type 2 error was very high and it would seem that an extremely low probability of imperfection should have been required of the monitor.

In fact, the monitor on Flight 903 did not fail (which is why we called its action "inappropriate" rather than "incorrect"): the problem was in the properties that it was required to monitor. Until now, we have implicitly considered demands to be events in the logical or physical world—but such events do not always announce themselves unambiguously: their detection may require delicate interpretation of sensors or indirect inference from other available data. In this case, the event to be monitored is malfunction of an avionics bus, and one way to detect this event is reception of faulty data delivered by the bus. And one way to judge whether data is faulty is by considering its physical plausibility. Here, roll angle rates of change greater than 40 degrees/second were considered implausible, and reception of sensor values in excess of this limit were what triggered the monitor. But, as noted, the airplane was stalled, the roll rates were real, and the inference of a bus fault was incorrect.

Designers choose the requirements for properties to be monitored; these properties, not the events they are intended to detect, are the demands that a monitor responds to and for which its probability of perfection is assessed. Through consideration of the possible costs of appropriate and inappropriate actions and inactions, designers may choose to fine-tune detection thresholds and adjust other elements in the specifications of properties to be monitored. They can also adjust the actions taken in response to violation of these properties. The consequences of these choices can far outweigh the perfection or otherwise of the monitor—hence, there is little point in requiring high probability of perfection in a monitor when the properties that it monitors are imperfect indicators of the event that it should respond to. Notice that this concern is not unique to monitors: it applies just the same to 1oo2 architectures.

It might at first seem reasonable to extend the probability of perfection for a monitor to include "perfection" of the monitored properties (relative the physical or logical events they are intended to identify). The problem is that perfection is an inappropriate notion to apply to these properties: they are internal models of an external reality and are subject to vicissitudes such as the fallibilities of sensors. The fidelity of a property to its intended event is therefore measured as reliability, not perfection.

For this reason, monitors and their required probabilities of perfection, the fidelity of the properties that they monitor, and the costs of the responses that they trigger and of the failures that they avert, should be developed in the context of a safety case, where the necessary assurances and tradeoffs can be made in an integrated manner. In the case of G-VATL, the property that was monitored by the FDC (low fuel in the engine tanks) had low fidelity with respect to the physical event of interest (malfunction of the fuel system) and was therefore prone to both Type 1 and Type 2 failures. To compensate for the latter, common mechanism was introduced between the operational system and the monitor, and that led to system failure. As we described earlier, a better design would have monitored the *distribution* of fuel: this would trigger much earlier and, modulo reliability of the sensors, would seem to have high fidelity. As noted, the cost of a false alarm is fairly modest in this system, and so we conclude that a properly constructed formal monitor would work well for this application: a modest and achievable probability of perfection would deliver a useful and credible claim to the safety case.

In the case of Flight 903, the property that was monitored (plausibility of data) had very low fidelity with respect to the event of interest (malfunction in an avionics bus) and the cost of a Type 2 failure was high (an automated bus reset). It seems unlikely that a monitor of higher fidelity could be constructed given the limitations of the subsystem concerned. We conclude that a monitored architecture was probably inappropriate for this application and that larger architectural revisions should have been considered, such as a fault-tolerant avionics bus, additional redundancy (e.g., checksums) to allow monitoring of end-to-end reliability, or more sophisticated fusion and diagnosis (e.g., analytical redundancy [47]) of available sensor data.

In summary, monitored architectures seem appropriate and effective for a usefully broad class of safety-critical systems. The monitor channel should evaluate properties taken directly from the system safety case and can be vastly simpler than the operational channel, as well as strongly diverse from it. Its simplicity should allow assessment of an attractively small probability of imperfection for the monitor channel, and the strong diversity between the two channels should allow separation of beliefs so that the complicating factor C employed in the 1oo2 case that led to the "cautionary" conclusion at the end of Section 2.3 is avoided. There are factors M_1 in (24) and M_2 in (25) that play arithmetically similar rôles to C, but these concern different events (failures due to common sensors and mechanisms shared between the operational and monitor channels) whose assessed probabilities can be reduced to small values by suitable design and assurance. (The problems with the A340 fuel management system example do not invalidate this conclusion: that was an unsuitable design.) A modest and plausible value such as 10^{-3} for the probability of perfection of the monitor, P_{B1}, then delivers a very useful reduction in the Type 1 failure rate for the system and a similar value should also be adequate for P_{B2} if the cost of a Type 2 failure can be controlled by suitable engineering.

5 Conclusion

We have considered systems in which a highly reliable software component or "channel" A operates in combination with a channel B that supports a credible claim of "perfection," meaning that it will never fail. The claim of perfection may be wrong, so we

speak of a "possibly perfect" channel characterized by a probability of imperfection. Our results for this model clarify, generalize, and extend an earlier result in [17].

The clarification lies in showing that there is *conditional independence* between the events "channel A fails on a randomly selected demand" and "channel B is imperfect." This means that the description of the aleatoric uncertainty is particularly simple: under conservative assumptions, the conditional system *pfd* is bounded by $p_A \times p_B$, where p_A is the probability that A fails on a randomly selected demand, and p_B is the probability that B is imperfect.

This contrasts with the usual approach involving channels for which claims of perfection cannot be made and only reliability is claimed; here, independence of channel failures *cannot* be asserted, and thus the two channel *pfd*s are not sufficient to characterize the model. Instead, consideration needs to be given to the "degree" of dependence of the failures of the two channels, and this is extremely difficult to incorporate into the model: in the Eckhardt and Lee [10] and Littlewood and Miller [16] models, for example, it involves knowing the covariance of the "difficulty functions" over the demand space. Typically, these are *not* known, and cannot easily be estimated.

The generalization lies in the treatment of epistemic uncertainty, which is shown to be the *sole* source of dependence in this model. Assessors must describe their beliefs about the two parameters of the model via the "belief" distribution $F(p_A, p_B)$ in order to obtain their unconditional probability of failure of the system on a randomly selected demand. Representing beliefs as a bivariate distribution may not be easy, but supposing that the distribution can be factored will usually not accurately represent the assessors' beliefs. We therefore proposed a conservative approach that is similar to that underpinning some "claim limit" approaches in safety cases, and that can be seen as one way of formalizing such arguments. It requires assessors to represent their beliefs in terms of just three quantities: the probability C of common faults that afflict both channels, the probability P_A^* of failure for the reliable channel A in the absence of such common faults, and the probability P_B^* of imperfection for the possibly perfect channel, again assuming the absence of common faults. The probability that the system fails on a randomly selected demand is then conservatively bounded by

$$C + (1 - C) \times P_A^* \times P_B^*.$$

The model described here seems much simpler than the classical one (with two reliable channels) in the treatment of both aleatory and epistemic uncertainty. We described the context of argument-based safety cases in which the epistemic probabilities C, P_A^*, and P_B^* should be formulated, and argued that credible and useful values can be constructed with modest enhancements to current development and assurance practices. Development of a channel that may claim perfection is most plausible when the channel is very simple, or is subjected to mechanically checked formal verification or, preferably, both. Fallibilities in formal verification are described in [36] and we argued that these can be accommodated within useful and credible values for P_B^*.

The earlier result has been extended by considering failures of commission (e.g., tripping a reactor when this is not needed), and then further extended by combining these results with a modification of the 1oo2 analysis to yield an analysis for monitored architectures: those, such as are proposed for aircraft, in which a reliable operational channel is monitored by a possibly perfect channel that can trigger higher-level fault recovery. These cases require extension of the statistical model from one based on probability of failure on demand to one based on rates of failure.

Like our first result, these extended results are relatively simple and require estimation of only a few parameters. The simplicity is achieved through conservative assumptions; hence our results are conservative also, possibly very much so. Nonetheless, plausible values for the parameters yield attractive conclusions.

The thrust of this paper has been about *assessment* of reliability of fault tolerant diverse software-based systems. The problem of assessing the reliability of such a system is often more difficult than the problem of *achieving* that reliability, particularly when the reliability requirements are very stringent. It is worth saying, then, that the asymmetric 1oo2 architecture considered here is one that has long been regarded as an attractive one for achieving reliability in certain applications (e.g., nuclear reactor protection systems), not least because it forces the reduction of excessive functionality, with its ensuing complexity, which are known enemies of dependability. That this architecture brings greater simplicity to the assessment process is an added bonus.

1oo2 architectures are most appropriate for systems that have a safe state; monitored architectures are one attempt to extend these benefits to systems that must continue operation in the presence of faults. Monitored architectures are recommended in the relevant aircraft guidelines [40, Table 5–2] as one way to achieve the high levels of dependability required for flight critical software. For example, it is suggested that a Level A system (one requiring the highest level of assurance) can be achieved by a Level C operational channel and a Level A monitor. Current revisions to these guidelines are augmenting its Table 5–2 with rules that allow calculation of acceptable alternatives. Our results can be seen as a way to provide formal underpinnings to those rules. We showed that risk due to failure of the operational channel can be significantly reduced by a monitor, but that the probability of imperfection in the monitor with respect to faults of commission must be carefully weighed against the danger and cost of unnecessary fault recovery. Furthermore, a perfect monitor is only as good as the properties that it monitors, so a large part of the challenge in developing effective monitors is to ensure that the properties chosen for monitoring are good ones.

Asymmetric 1oo2 systems and monitored architectures can be seen as ways to implement and exploit ideas that have long been known in the fields of fault tolerance and security: for example, fail-stop processors [38], safety kernels [35, 46], detectors and correctors [1], and security by execution monitoring [39]. All of these use 1oo2 comparison or some form of monitoring to avoid transitions to hazardous states. The utility of these architectures in the construction of critical systems is well known; our contribution is to show how the particular form of these architectures (where the monitor can support a claim of probable perfection) enables assessment of the reliability achieved.

As we have indicated at various points in the paper, assessment at the epistemic level in models of these kinds is not easy (although, as we have also noted, it is considerably easier than for models in which claims about reliability must be made for both channels). Our treatment makes assumptions that may be very conservative, and still the parameter C, for example, presents great difficulty for an assessor. More work is needed on ways of easing the task of assessors in using our approach: we briefly mentioned some recent results that avoid the problem of assessing C by requiring the assessors instead to provide assessments about their marginal distributions for pfd_A and pnp_B. We intend to investigate alternative approaches to these problems.

Other work for the future could involve extending the scope of the models. For example, it will be interesting to extend the analysis of monitored architectures to

include recovery mechanisms, and possibly to consider a basis for certification of systems that employ adaptive control or other techniques that are currently considered beyond the pale.

Acknowledgements

We are indebted to the Associate Editor, and three reviewers, for their careful readings of the original manuscript and extensive helpful suggestions that greatly improved the final paper.

We would also like to thank the following colleagues for valuable discussions about earlier versions of this work: Peter Bishop, Robin Bloomfield, Alan Burns, Bruno Dutertre, Pat Lincoln, Paul Miner, Peter Popov, Andrey Povyakalo, Bob Riemenschneider, N. Shankar, Lorenzo Strigini, David Wright, and Bob Yates.

The first author's work was supported by the INDEED project, funded by the UK Engineering and Physical Sciences Research Council; and by the DISPO project, funded by British Energy, NDA (Sellafield, Magnox North, Magnox South), AWE and Westinghouse under the CINIF Nuclear Research Programme (the views expressed in this Report are those of the authors and do not necessarily represent the views of the members of the Parties; the Parties do not accept liability for any damage or loss incurred as a result of the information contained in this Report).

The second author was supported by NASA cooperative agreements NNX08AC64A and NNX08AY53A, and by National Science Foundation grant CNS-0720908.

References

[1] Anish Arora and Sandeep S. Kulkarni. Designing masking fault tolerance via nonmasking fault tolerance. *IEEE Transactions on Software Engineering*, 24(6):435–450, June 1998.

[2] *In-Flight Upset Event, 154 km West of Learmonth, WA, 7 October 2008, VH-QPA Airbus A330-303.* Australian Transport Safety Bureau, March 2009. Reference number AO-2008-070 Interim Factual and Interim Factual No. 2, available at `http://www.atsb.gov.au/publications/investigation_reports/2008/aair/ao-2008-070.aspx`.

[3] *Licensing of Safety Critical Software for Nuclear Reactors: Common Position of Seven European Nuclear Regulators and Authorised Technical Support Organizations.* AVN Belgium, BfS Germany, CSN Spain, ISTec Germany, NII United Kingdom, SKI Sweden, STUK Finland, 2007. Available at `http://www.bfs.de/de/kerntechnik/sicherheit/Licensing_safety_critical_software.pdf`.

[4] Howard Barringer, David Rydeheard, and Klaus Havelund. Rule systems for runtime monitoring: From EAGLE to RULER. In *Runtime Verification (RV 2007)*, Volume 4839 of Springer-Verlag *Lecture Notes in Computer Science*, pages 111–125, Springer-Verlag, Vancouver, British Columbia, Canada, March 2007.

[5] Peter Bishop and Robin Bloomfield. A methodology for safety case development. In *Safety-Critical Systems Symposium*, Birmingham, UK, February 1998. Available at `http://www.adelard.com/resources/papers/pdf/sss98web.pdf`.

[6] *Statistical Summary of Commercial Jet Aircraft Accidents, Worldwide Operations, 1959–2009.* Boeing Commercial Airplane Group, Seattle, WA, July 2010. Published annually by Boeing Airplane Safety Engineering, available at `http://www.boeing.com/news/techissues/pdf/statsum.pdf`.

[7] Ricky W. Butler and George B. Finelli. The infeasibility of experimental quantification of life-critical software reliability. *IEEE Transactions on Software Engineering*, 19(1):3–12, January 1993.

[8] *CAST Position Paper 24: Reliance on Development Assurance Alone when Performing a Complex and Full-Time Critical Function.* Certification Authorities Software Team (CAST), March 2006. Available from `http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/`.

[9] Dave E. Eckhardt, Alper K. Caglayan, John C. Knight, Larry D. Lee, David F. McAllister, Mladen A. Vouk, and John P. J. Kelly. An experimental evaluation of software redundancy as a strategy for improving reliability. *IEEE Transactions on Software Engineering*, 17(7):692–702, July 1991.

[10] Dave E. Eckhardt, Jr. and Larry D. Lee. A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Transactions on Software Engineering*, SE-11(12):1511–1517, December 1985.

[11] *System Design and Analysis.* Federal Aviation Administration, June 21, 1988. Advisory Circular 25.1309-1A.

[12] Klaus Havelund and Grigore Rosu. Efficient monitoring of safety properties. *Software Tools for Technology Transfer*, 6(2):158–173, August 2004.

[13] Tim Kelly. *Arguing Safety—A Systematic Approach to Safety Case Management*. PhD thesis, Department of Computer Science, University of York, UK, 1998.

[14] J. C. Knight and N. G. Leveson. An empirical study of failure probabilities in multi-version software. In *Fault Tolerant Computing Symposium 16*, pages 165–170, IEEE Computer Society, Vienna, Austria, July 1986.

[15] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1):96–109, January 1986.

[16] B. Littlewood and D. R. Miller. Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering*, 15(12):1596–1614, December 1989.

[17] Bev Littlewood. The use of proof in diversity arguments. *IEEE Transactions on Software Engineering*, 26(10):1022–1023, October 2000.

[18] Bev Littlewood, Peter T. Popov, Lorenzo Strigini, and Nick Shryane. Modelling the effects of combining diverse software fault detection techniques. *IEEE Transactions on Software Engineering*, 26(12):1157–1167, December 2000.

[19] Bev Littlewood and Andrey Povyakalo. Conservative reasoning about epistemic uncertainty for the probability of failure on demand of a 1oo2 software-based system in which one channel is "possibly perfect". Technical report, Centre for Software Reliability, City University, London, UK, January 2010.

[20] Bev Littlewood and Andrey Povyakalo. On claims for the perfection of software. Technical report, Centre for Software Reliability, City University, London, UK, January 2010.

[21] Bev Littlewood and John Rushby. On the nature of software assurance. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, 2010. In preparation.

[22] Bev Littlewood and Lorenzo Strigini. Validation of ultrahigh dependability for software-based systems. *Communications of the ACM*, pages 69–80, November 1993.

[23] Bev Littlewood and David Wright. The use of multi-legged arguments to increase confidence in safety claims for software-based systems: a study based on a BBN analysis of an idealised example. *IEEE Transactions on Software Engineering*, 33(5):347–365, May 2007.

[24] E. Lloyd and W. Tye. *Systematic Safety: Safety Assessment of Aircraft Systems*. Civil Aviation Authority, London, England, 1982. Reprinted 1992.

[25] J. May, G. Hughes, and A. D. Lunn. Reliability estimation from appropriate testing of plant protection software. *IEE/BCS Software Engineering Journal*, 10(6):206–218, November 1995.

[26] *Development Guidelines for Vehicle Based Software.* The Motor Industry Software Reliability Association (MISRA), Nuneaton, UK, January 2001. PDF Version 1.1.

[27] *Safety Recommendations A-98-3 through -5.* National Transportation Safety Board, Washington, DC, January 1998. Available at `http://www.ntsb.gov/Recs/letters/1998/A98_3_5.pdf`.

[28] William L. Oberkampf and Jon C. Helton. Alternative representations of epistemic uncertainty. *Reliability Engineering and System Safety*, 85(1–3):1–10, 2004.

[29] Anthony O'Hagan, Caitlin E. Buck, Alireza Daneshkhah, J. Richard Eiser, Paul H. Garthwaite, David J. Jenkinson, Jeremy E. Oakley, and Tim Rakow. *Uncertain Judgements: Eliciting Experts' Probabilities.* Wiley, 2006.

[30] *Engineering Safety Management (The Yellow Book), Volumes 1 and 2, Fundamentals and Guidance, Issue 4.* Rail Safety and Standards Board, London, UK, 2007. Available from `http://www.yellowbook-rail.org.uk/site/the_yellow_book/the_yellow_book.html`.

[31] *DO-178B: Software Considerations in Airborne Systems and Equipment Certification.* Requirements and Technical Concepts for Aviation, Washington, DC, December 1992. This document is known as EUROCAE ED-12B in Europe.

[32] *DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations.* Requirements and Technical Concepts for Aviation, Washington, DC, November 2005. Also issued as EUROCAE ED-124 (2007).

[33] Sheldon M. Ross. *Stochastic Processes.* Wiley, New York, NY, 1996.

[34] J. C. Rouquet and P. J. Traverse. Safe and reliable computing on board the Airbus and ATR aircraft. In *Safety of Computer Control Systems (SAFECOMP '86)*, Published by Pergamon for the International Federation of Automatic Control, IFAC, Sarlat, France, July 1986.

[35] John Rushby. Kernels for safety? In T. Anderson, editor, *Safe and Secure Computing Systems*, chapter 13, pages 210–220. Blackwell Scientific Publications, 1989. (Proceedings of a Symposium held in Glasgow, October 1986).

[36] John Rushby. Software verification and system assurance. In Dang Van Hung and Padmanabhan Krishnan, editors, *Seventh International Conference on Software Engineering and Formal Methods (SEFM)*, pages 3–10, IEEE Computer Society, Hanoi, Vietnam, November 2009.

[37] *Air Traffic Services Safety Requirements, CAP 670.* Safety Regulation Group, UK Civil Aviation Authority, June 2008. See Part B, Section 3, Systems Engineering SW01: Regulatory Objectives for Software Safety Assurance in ATS Equipment; Available at `http://www.caa.co.uk/docs/33/cap670.pdf`.

[38] R. D. Schlichting and F. B. Schneider. Fail-stop processors: An approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems*, 1(3):222–238, April 1983.

[39] Fred Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, February 2000.

[40] *Aerospace Recommended Practice (ARP) 4754: Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* Society of Automotive Engineers, November 1996. Also issued as EUROCAE ED-79.

[41] *Report on the incident to Airbus A340-642, registration G-VATL en-route from Hong Kong to London Heathrow on 8 February 2005.* UK Air Investigations Branch, 2007. Available at `http://www.aaib.gov.uk/publications/formal_reports/4_2007_g_vatl.cfm`.

[42] *The Use of Computers in Safety-Critical Applications: Final Report of the Study Group on the Safety of Operational Computer Systems.* UK Health and Safety Commission, 1998. Available at `http://www.hse.gov.uk/nuclear/computers.pdf`.

[43] *Safety Assessment Principles for Nuclear Facilities.* UK Health and Safety Executive, Bootle, UK, 2006 edition, version 1 edition. Available at `http://www.hse.gov.uk/nuclear/saps/saps2006.pdf`.

[44] *Health and Safety at Work etc. Act.* UK Health and Safety Executive, 1974. Available at `http://www.hse.gov.uk/legislation/hswa.htm`; guidance suite at `http://www.hse.gov.uk/risk/theory/alarp.htm`.

[45] *Defence Standard 00-56, Issue 4: Safety Management Requirements for Defence Systems. Part 1: Requirements.* UK Ministry of Defence, June 2007. Available at `http://www.dstan.mod.uk/data/00/056/01000400.pdf`.

[46] Kevin G. Wika and John C. Knight. On the enforcement of software safety policies. In *COMPASS '95 (Proceedings of the Tenth Annual Conference on Computer Assurance)*, pages 83–93, IEEE Washington Section, Gaithersburg, MD, June 1995.

[47] Alan S. Willsky. A survey of methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611, November 1976.

The Authors

Bev Littlewood has degrees in mathematics and statistics, and a PhD in statistics and computer science; he is a Chartered Engineer, and a Chartered Statistician. He has worked for more than 30 years on problems associated with the dependability of software-based systems, and has published many papers in international journals and conference proceedings and has edited several books. His technical contributions have largely focused on the application of rigorous probabilistic and statistical techniques in software systems engineering. In 1983 he founded the Centre for Software Reliability (CSR) at City University, London, and was its Director until 2003. He is currently Professor of Software Engineering in CSR. From 1990 to 2005 he was a member of the UK Nuclear Safety Advisory Committee. He is a member of IFIP Working Group 10.4 on Reliable Computing and Fault Tolerance, of the UK Computing Research Committee, and is a Fellow of the Royal Statistical Society. He is on the editorial boards of several international journals. In 2007 he was the recipient of the IEEE Computer Society's Harlan D Mills Award.

John Rushby received B.Sc. and Ph.D. degrees in computing science from the University of Newcastle upon Tyne in 1971 and 1977, respectively. He joined the Computer Science Laboratory of SRI International in 1983, and served as its director from 1986 to 1990; he currently manages its research program in formal methods and dependable systems. His research interests center on the use of formal methods for problems in the design and assurance of secure and dependable systems. Prior to joining SRI, he held academic positions at the Universities of Manchester and Newcastle upon Tyne in England.

Dr. Rushby is a former associate editor for Communications of the ACM, IEEE Transactions on Software engineering, and Formal Aspects of Computing. He was recently a member of a National Research Council study that produced the report "Software for Dependable Systems: Sufficient Evidence?".