The impact of diversity upon common mode failures

Bev Littlewood
Centre for Software Reliability
City University
Northampton Square
London EC1V 0HB

Abstract

Recent models for the failure behaviour of systems involving redundancy and diversity have shown that common mode failures can be accounted for in terms of the variability of the failure probability of components over operational environments. Whenever such variability is present, we can expect that the overall system reliability will be less than we could have expected if the components could have been assumed to fail independently. We generalise a model of hardware redundancy due to Hughes [Hughes 1987], and show that with *forced diversity*, this unwelcome result no longer applies: in fact it becomes theoretically possible to do better than would be the case under independence of failures.

Acknowledgement

The work reported here has been partially supported by the following projects on system dependability within the Centre for Software Reliability: DATUM (funded by the DTI/EPSRC Safety Critical Systems Research Programme, grant GR/H89944), SHIP (under the EU Environment programme, EV5V 103) and PDCS2 (an EU ESPRIT Basic Research project, 6362).

1 Introduction

Notions of redundancy and diversity are ubiquitous in the design of systems which have demanding safety or reliability requirements. Whilst there is clear evidence that these approaches can bring benefits when compared with unitary systems, these benefits can be difficult to quantify. In particular, it is usually difficult to measure the overall system reliability or safety when redundancy and/or diversity plays a rôle in system design.

At the very simplest level, where components can be replicated *and their failures in operation can be assumed to be statistically independent*, we know that we can build arbitrarily reliable systems with arbitrarily unreliable components. Quite elementary mathematics will allow the system reliability to be computed in terms of component reliabilities. Unfortunately, even in this simple case of *redundancy*, such results are little more than mathematical curiosities, since the assumptions of complete independence are rarely (probably never) justified. We are therefore in practice forced into estimating the effects of common mode failures. At a higher level of sophistication, the use of *design diversity* as a means of masking failures caused by design faults in complex systems introduces, as we shall see, novel and perhaps intractable problems. In particular, there is overwhelming evidence that here also it is not possible to ensure statistical independence in the failure processes of the different versions, and so once again the effect of common mode failures must be taken into account in the assessment of such a system.

For the most part, until quite recently, evidence for the practical efficacy of redundancy and diversity, taking account of the possible common mode failures, relied largely upon quite sparse empirical evidence. This relative paucity of data on common mode failures resulted in a concentration upon qualitative guidance on the design and operation of redundant systems, with only cursory treatment of a quantitative view at system level [Bourne, Edwards et al. 1981]. Such a qualitative view has admittedly had considerable success in *achieving* reliability: we know from operational experience that some systems exhibit extraordinarily high reliability. But it does not help those responsible for the assessment of the reliability and safety of particular systems *before* they are put into operation.

A typical pragmatic approach to the problem of quantification involves the β-factor - the probability that if a failure occurs in one channel other channels are also failed due to a common cause. Bourne *et al* [*op cit*] show the effect of common mode failures upon system reliability. The efficacy of a redundant design is exemplified via calculations based on a 2-out-of-3 system: the system failure rate is plotted against β-factor for various values of the component reliability (probability of failure). An important feature is the part of these plots that relates to those β-factor values for which the authors claim that some data exist - between 0.07 and 0.4. The authors give the example of a 2-out-of-3 system for which a system failure probability of better than 10^{-3} is required. They show that taking account of common mode failure possibilities one would realistically need to have a component failure probability of better than 5×10^{-3}; this compares with the component failure probability of only 1.8×10^{-2} that would suffice if there were independence between the three components. The interest of such calculations for us is that they show the seriousness of the problem when faced with β-factors of the magnitude that one could, it appears, reasonably expect. Most

importantly, they show that we must be able to estimate accurately the *actual degree of dependence* between the different component failure processes before we can make accurate estimates of the overall system reliability or safety.

There is an assumption in all this work involving the β-factor that such a factor alone *can* capture the effect of common mode failure sources on the final system reliability - essentially it is being assumed that failures due to common causes are simply related to single channel failures via the β-factor. Is this reasonable? It could be argued that the techniques used to increase reliability generally are also successful at finding those faults that are common between channels. On the other hand, a more conservative view might be that common faults are different in kind from single channel faults. In the worst case then it might be that putting a lot of effort into increasing channel reliabilities does not significantly improve that part of the total unreliability that is due to common modes - essentially the β-factor is not constant, and the proportion of common mode failures increases as component reliabilities increase. There seems little hard evidence on this issue. Even in the case of software there is little agreement on the question. It is often argued that the vulnerability to design faults in software is influenced by how hard one has tried to eliminate these (extensiveness of testing, etc), but there is hardly any quantitative evidence and we know even less about the effect of common faults in diverse software versions. This is an area that will repay further research effort, not least to resolve open questions of relative cost-effectiveness of different design approaches.

The numerical example above illustrates the first important lesson here: with a realistic assumption about the possibility of common cause failures, the system reliability can be *dramatically* less than we could expect under the (incorrect) assumption of independence. The assumptions that go into these results, it should be said, are not particularly conservative; indeed, they are based upon real data, albeit of a scanty nature. In [Bourne, Edwards et al. 1981] *hardware* systems are being discussed throughout, but the picture is very much the same for software systems and there is some evidence that things may even be worse. Experiments have been conducted [Eckhardt, Caglayan et al. 1991; Knight and Leveson 1986] in which many diverse versions have been developed and tested on many millions of input cases in order to detect common faults. Dramatic differences were observed between actual achieved system reliability for, say, a 2-out-of-3 system, and the nominal reliability that could be expected under an assumption of independence.

In [Knight and Leveson 1986] there were several instances in which many of their 27 versions shared common faults. Thus *which* three versions were chosen for a 2-out-of-3 system would determine the behaviour of the system when an input triggered such version failures - the number of failed versions in the triplet could be none, one, two or three. This brings us to the second cautionary observation here - that there could be great uncertainty in the actual results to be obtained from a system in which there is the possibility for common mode failures, even if we were able to be confident of the benefits that might be achieved *on average*. If we are not in a position to measure what has *actually been achieved* with the particular application of redundancy or diversity, it seems vital that, at least in a safety-critical context, only conservative assumptions should be allowed in claims about the reliability of such a system.

Clearly there are large gaps in our understanding of some of the basic issues here. We know that we would like to have independence of the failure behaviour of the components in a redundant or diverse system, because this would allow us to carry out

quite simple calculations to determine system reliability. If we cannot claim independence, then we need to estimate the degree of dependence achieved for the particular system under examination in order to compute its reliability. Unfortunately, means of using information about system design in order to estimate this dependence are very poor; and direct empirical evidence of common failures is, by its nature, extremely sparse. The poor general understanding is illustrated well in the literature concerning software diversity for fault tolerance, where until recently it was common to use words like 'independent' and 'diverse' quite loosely: for example, it was said that 'independent development' of 'diverse' versions was a means to obtaining 'independent' failure behaviour in the versions. Before we can develop theories and models that will allow us to predict system reliability in the presence of common mode failures, we need to have a basic understanding of these fundamental concepts and their relationships. We need to answer questions such as: what *is* diversity? are *these* designs more diverse than *those*? *how* diverse are these two designs? what diversity can I expect by allowing designers complete freedom, but forbidding their communication with one another? can I *force* greater diversity by insisting on 'doing things differently'? how does version *diversity* impact upon version *failure behaviour*? In the past few years, formal models have been proposed that start to answer some of these questions.

2 The effect of environmental variability upon common mode failures

2.1 The models of Hughes and of Eckhardt and Lee

The first serious attempts to construct a formal model for common mode failures were due to Hughes [Hughes 1987] and Eckhardt and Lee [Eckhardt and Lee 1985], who seem to have worked independently to arrive at similar models to characterise the nature of failure dependence. Hughes discusses the problem in the framework of hardware reliability, whereas Eckhardt and Lee (E&L) treat failures due to design faults in software. The later work of Littlewood and Miller [Littlewood and Miller 1989] extends the model of Eckhardt and Lee, and the main objective of the present paper is to show that there is a similar extension to the Hughes model for hardware.

The key notion in both the Hughes and E&L models is *variability*. In the Hughes model, there is variability, from one operating environment to another, in the probability that a particular type of component will fail. By 'operating environment', Hughes has in mind a very general formulation that includes, for example, maintenance policy. If we were to place more than one similar component in a redundant architecture, in order to try to improve the reliability over what we could expect from a single component, it is the nature of the distribution of this variability of failure probability that determines how successful we would be. Very informally, the idea here is as follows. Let us imagine that we have built a system from two similar components, and the system works successfully if at least one component works. As an example consider an emergency cooling system comprising two similar pumps: a demand upon the system is satisfied if at least one of the pumps starts up successfully when the demand occurs. Now we observe that *one* of the pumps has indeed failed: what do we think is the chance that the other will fail? Under a naive assumption of *independence* of failure behaviour between the two components, the probability that the second pump will also fail is merely the *marginal* probability of failure of a pump. If this assumption of independence were correct, then, we could expect a significantly greater reliability

for the 1-out-of-2 system than for a single component alone: in fact a probability of failure p^2 instead of the single component probability of failure p.

In the Hughes model, we reason that, since the first pump has failed, *this is probably a stressful environment for all pumps of this type*, and thus the second component will be more likely to fail. At its worst, in this model each environment could have the property that either all components fail, or all components work - in which case knowledge that the first component had failed would mean it was certain that the second component would fail, and thus so would the 1-out-of-2 *system* (Figure 1b).

In general, whenever there is variation of stressfulness between environments (and it seems hard to conceive of circumstances where this is not the case) the 1-out-of-2 system will have a reliability less than that given by the independence assumption: the probability of system failure will lie somewhere between p and p^2.

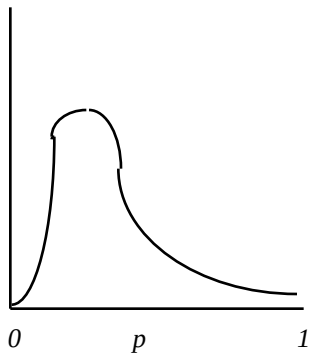

Figure 1a. The most general formulation, in which there is 'continuous variation' of stressfulness over environments.

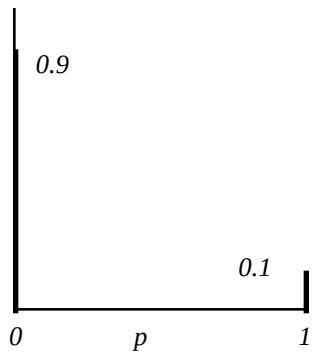

Figure 1b. The case when, with probability *0.1*, all components of the same type are certain to fail. In this case the two-pump system has exactly the same probability of failure as a single pump and redundancy brings no advantage.

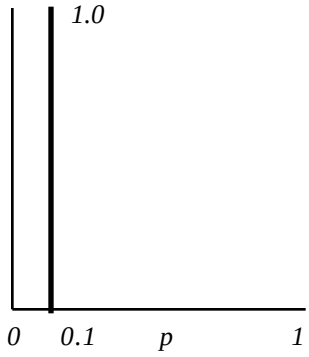

Figure 1c. The case of completely independent failures. Here the two-pump system has a probability of failure *0.1*×*0.1*, giving the greatest benefit from redundancy in this scenario.

To be a little more formal about the model, we need to state carefully the nature of the uncertainty in the various statements about probabilities. In the Hughes model, let $P(fail|e)$ represent the probability that a particular type of component fails in a particular environment, e. We could estimate $P(fail|e)$ by putting many such components to work in this environment, and calculating the proportion that fail. Thus $P(fail|e)$ can be thought of as a conditional probability - it is the probability that a (randomly chosen) component will fail *given that it is in operating environment* e. Now there are many (often an infinite number of) environments, and the selection of an environment is itself a random process. If we have a mechanism that selects environments for us, this induces a probability distribution on $P(fail|e)$: say

$F(p) \equiv P(P(fail|e) < p)$. It is the shape of this distribution, and in particular its variance, that determines the degree of success that will come from building redundant systems from such components. The worst case, mentioned above, is one where all the probability mass of F(p) is concentrated at 0 and 1, when no benefit at all is gained from redundancy, since this guarantees that *all* failures of components will be coincident ones. The case of *independent* failures is that where all the probability mass is concentrated at some point between 0 and 1 (Figure 1c). In reality something between these two extremes will be the case, and it would be reasonable to believe that the probability will be spread over the whole interval (0,1) - in which case the precise shape of this distribution will determine the efficacy of a redundant architecture (Figure 1a).

It is important to stress here the very different sources of uncertainty that are represented by the different probabilities. When we talk of the probability $p = P(fail|e)$, we are referring to the uncertainty of failure behaviour between different components of the same type in the same environment: even if we kept the environment completely fixed, there would be uncertainty about the failure behaviour of individual components. Thus we would not know for certain how different pumps would behave under the same demand. The distribution $F(p)$, on the other hand, deals with the variation of this p over different randomly selected environments. It concerns the differing propensity of this type of component to succumb to different environments. The variance of this distribution is the key: the greater this is, the greater the propensity to failure commonality and thus the further we shall be from the 'independence' ideal.

In the E&L model the variation of 'environment' is replaced by variation of 'inputs', and this is somewhat more plausible in this software context. Also, instead of redundancy (involving similar components), we here have *diversity* - different *versions* of a program involving different designs. The informal arguments here are somewhat similar to those of Hughes in the hardware redundancy case. It is assumed that there is a large space of possible inputs, and that execution of the program in a particular environment involves the selection of an input via a probability distribution over this space. This probability distribution therefore represents the 'operational' environment[1], and will vary from one environment, or type of operation, to another. It is variation *within* such an environment, of the 'difficulty' of an input, that is the key to the modelling of failure dependence. Consider, for simplicity, the case of a 1-out-of-2 system, involving two diverse versions. If an input is selected, and version A fails to execute that input correctly, what is the chance that version B will also fail, i.e. that the *system* will fail? As in the Hughes model, the probability is *greater* than the marginal probability of failure of a single version, which would be the case under a naive assumption of failure independence. Here the reasoning is that the failure of A suggests that the input was probably a 'difficult' one, and thus the chance of B failing is greater than it otherwise would be. The underlying informal idea is that designing software involves thinking of solutions to different problems, represented by different sets of inputs, and that some of these problems represent greater intellectual challenges (and thus scope for human error) than others. Consider, as an example, the case of a fly-by-

[1] There is a potential confusion of terminology between the Hughes and Eckhardt and Lee models. In E&L, 'input' plays the role that 'environment' plays in Hughes. However, it is usual in the software context also to talk of the 'environment' in which a program operates: this is the space of all inputs together with the probabilistic mechanism for selecting sequences of inputs during operation.

wire aircraft: the inputs being received during a landing in severe wind-shear would be intrinsically harder to respond to correctly (i.e. would be intrinsically harder for the designers of the different versions to program a correct response to) than those coming from straight and level flight in perfect conditions.

Once again, it is necessary to spell out carefully the precise nature of the randomness in order to understand what the different probabilities mean. For a particular input x, the probability of a program failing is (in the E&L notation) $\theta(x)$. This can be thought of as the proportion, of all programs that *could* be written, that would fail on this input. That this probability is a function of x is the key to the model of dependency: if this probability is large (i.e. there is a tendency for many programs to fail), then x is a 'difficult' input. In this model, the idea of operational environment plays a different role from that in the Hughes model. Here the operational environment is represented by the probability distribution over the different inputs, x, which in turn induces a probability distribution on the probabilities of failure $\theta(x)$; in E&L notation we have $G(\theta) = P(\theta(x) < \theta)$. As before, it is the variance of this distribution that plays an important role; the greater the variance, the greater the deviation from the kind of failure behaviour that we would expect if the failures of the different versions were to occur independently.

2.2 Discussion

Both models embody carefully thought out representations of key notions such as 'similarity' and 'independence'. By itself, this is a valuable service. In the software literature, in particular, these words have been used heretofore in a rather cavalier fashion: thus builders of fault-tolerant systems have said that the different versions were developed 'independently', in order to create 'independent' versions that would fail 'independently'! In E&L (and in Hughes, although there things are stated less explicitly) these different roles of independence (and thus, by inference, dependence) are defined rigorously. First, there is the independence of the developments of the different versions. This arises formally in E&L via random independent selection of the program versions from the population of all programs - the distribution used for this selection can be thought of as in some way characterised by the nature of the application problem and perhaps of the development methods available. Thus in the model, programs can be thought of as 'similar' in the sense that they can be regarded as independent, *identically distributed* objects, so capturing the idea that the diversity arises 'naturally' (we shall see later how this idea can be extended to embrace the notion of *forced* diversity). Secondly, there is *conditional* independence of failure behaviour of the versions: specifically, *for any given input* x the different versions fail independently. Finally, there is the subtle implication of all this: there is *not* independence in the *unconditional* failure processes of the different versions. That is, for a randomly selected input (i.e. for a future unknown input), the versions fail dependently.

For all their mathematical similarity, however, there are some important differences of interpretation in the two models. Informally, I hope to show that there is 'more similarity' between components in the Hughes model than there is between versions in E&L, and the components in the Hughes model behave 'more randomly'.

The Hughes model is concerned with *redundancy* in hardware. It treats the case where nominally similar components - identical in design - are placed in the same 'environment'. In the E&L model, on the other hand, the different 'components' -

program versions - are *known* to be different in kind, i.e. in design. The purpose of design diversity is precisely to make them differ in the hope that differences in design will lessen the possibility of simultaneous failure, due to design faults, under particular circumstances.

In the Hughes model, each component has a possibility of failure in the selected environment: in the case of the pumps, for example, the 'environment' may be represented by the physical conditions under which a demand for start-up occurs. Within a particular environment, the failures of different components are independent (i.e. they are *conditionally* independent). The randomness, or uncertainty as to whether a failure of a particular component *will* occur, arises at least partly as a result of unobserved 'natural' differences in the physical make-up of the components.

An issue that is not addressed in the Hughes model, and to which I shall return in more detail in the next section, concerns the possibility of variation in failure behaviour *between* components. It seems clear that a component will in general not always behave the same when presented repeatedly with the same environment: for example it may fail today in an environment in which it worked successfully yesterday. This means that 'probability of failure' is a meaningful concept not only in terms of the proportion of all components that might fail when each is presented singly with a particular environment, as considered by Hughes, but also for a single component when repeatedly presented with the same environment. Thus, as an example, we could use a geometric distribution for the probability that a component used repeatedly, and independently, in the same environment will succeed n times, and fail on the $(n+1)th$ trial. We could even, in the case where the probability of failure of a particular component in a particular environment is unknown, carry out statistical inference as a result of seeing a particular number of successful operations of the component in the environment[2]. Most importantly, it is sensible to ask whether, for a particular environment, all components will have the same probability of failure: it seems clear that, in practice, this will not be the case. If so, the probability of failure discussed by Hughes, $P(fail|e)$, can then be thought of as an average, over the population of all components, of these individual component probabilities of failure in the environment e (see section 3). The precise nature of this average will depend upon the nature of the (random) selection mechanism for choosing components: in many cases it will be realistic to assume equally-likely selection.

In contrast to the Hughes model, in E&L each version, when presented with an input, has a possibility of failure, but whether or not a failure actually occurs is, in a sense, deterministic: thus a version that fails on a particular input will *always* fail on that input. The indeterminacy here arises only because we do not know, *a priori*, whether a particular version is prone to fail on a new input. The 'probability of failure' here can only be thought of as the proportion of all programs that fail on that input; in particular it does not make sense in this model, in contrast to Hughes', to talk of the probability of failure of a *particular* version on a *particular* input (such a 'probability' is always either *1* or *0*). That is, the uncertainty arises in E&L only via the process of selection of the version: our uncertainty arises from our ignorance as to the properties - in particular failure propensities over the input space - of the version. We can imagine

---

[2] This observation does not require any notion of repairability: merely seeing n successes in n trials allows us to infer something about the probability of failure, at least in the Bayesian framework. If, in addition, we allow the component to be restorable to its original state by 'repair', we can imagine a series of Binomial trials which allow the usual statistical inference, Bayesian or classical, for the probability of failure. Of course, such inference will rarely be possible in practice.

(although it is a practical impossibility) knowing the outcomes (failure or success) of *all* programs on *all* inputs, whereupon the model becomes completely deterministic; there is no such completely deterministic case in the Hughes model.

Notice that the Hughes model does not rule out the possibility of a common design fault causing *all* components to fail whenever they are required to operate in a particular environment. This would be represented by there being non-zero probability associated with the point *p=1* in Figure 1. However, it is possible that even a design fault will not always be certain to cause a particular component to fail each time a certain environment is encountered, so some of the probability associated with the interval (*0,1*) may also refer to design faults. Equally, two components with the same design fault may not always fail together (there may be unexplained variation between them that determines whether or not the design fault results in a failure in a particular environment). Once again, this contrasts with the case of software, where it is conventional to think of a common design fault in two versions as corresponding to a set of inputs, all of which would fail both versions[3].

The distinctions being made here are quite subtle, for example between nominally similar components with hidden physical differences, on the one hand, and deliberately different versions where these difference have unknown consequences, on the other hand. However, important practical differences arise when we try to carry out statistical inference on these models. In the Hughes case, it is likely that there will be data available from a comparatively large number of components, operating in a relatively small number of environments. In the case of E&L, there is likely to be data from only a very small number of versions, but the number of possible inputs that could be observed will be large. Even in the case of software diversity experiments [Knight and Leveson 1986], involving 'small' problems, it is expensive to generate many versions; for real problems it is likely that it would only be possible to study the two or three versions that would eventually be used in a fault-tolerant architecture in the final system. This clearly presents serious problems of inference, since the important probabilities in E&L are defined only over the population of *all programs*.

2.3 Littlewood and Miller model

Both the Hughes and E&L models, in their different applications to hardware and software, address the problem of common mode failures in circumstances when there is a high level of 'similarity' in the redundancy or diversity that is present in the system. Thus in Hughes, although not stated explicitly, it seems clear that it is the intention to model the situation where components of *identical* type - i.e. design - are placed in a redundant structure. In E&L, the different program versions will have different designs, but these differences will have arisen, merely willy-nilly, as a result of the use of different teams of designers being given the *same* specification - the different programs can be regarded as independent, identically distributed random objects. Most importantly, there is no attempt to force diversity by specifying that the different teams do things in different, particular, ways; diversity merely arises as a result of random selection[4].

---

[3] This conventional assumption is rather strong, even for software (and is not required within the E&L model). One could imagine, for example, inputs on the 'edge' of the fault region where one program fails and another succeeds as a result of other differences between them.

[4] The detailed mechanism that is operating here is not modelled directly in E&L. Presumably it

Littlewood and Miller [Littlewood and Miller 1989] generalise the E&L model to represent the situation where diversity of design is forced upon the different versions. Practical examples of such forced diversity might include stipulating that the different development teams use different programming languages, different testing methods, etc. Clearly, some discretion would need to be exercised in practice over the extent to which diversity - and thus particular design solutions - would be imposed on the developers. Intuitively, the idea here would be to try to force sufficient 'deliberate' diversity as to decrease the tendency for coincident failures, without introducing novel opportunities for commonality via the inevitable extra commonality in the high-level design.

For the simple 1-out-of-2 case, the L&M model assumes that there are two different methodologies A and B which will be used to develop the two different versions. We would like it to be the case that when there is a large probability of the type-A program failing, there is a *small* probability of the type-B program failing - and vice-versa. Specifically, it is assumed that the type-A program will have probability $P_A(x)$ of failing on an input x, and the type-B program a probability $P_B(x)$. As before, these probabilities can be thought of as the proportions of programs, from the populations of all programs that could be written using the two methodologies, that would fail to execute x correctly. The joint distribution $G(p_A, p_B) = P(P_A(x)<p_A, P_B(x)<p_B)$ then determines the efficacy of the fault-tolerant system. Just as, in the other two models, it is the variance of the univariate distribution that plays the key role, here it is the correlation coefficient that is the key. It can be shown that if this is *negative*, then we shall do better even than what would be achieved under the assumption of complete independence of failure behaviour of the versions. Even in the case of positive correlation, the E&L result can be shown to be a worst case - essentially it is the case where the distribution $G(p_A, p_B)$ in the (p_A, p_B) plane is concentrated entirely on the line of unit slope.

In the next section we use the mathematical framework of the L&M model to generalise the Hughes model before going on, in the following section, to represent the situation in which components with known-different designs are used in a redundant architecture.

3 A generalised Hughes model

We begin with some notation that closely follows that in [Littlewood and Miller 1989]. There is a population of all possible components

$$\boldsymbol{C} = \{c_1, c_2, \ldots\} \tag{1}$$

from which a particular component will be randomly selected to be used in a particular context. Thus the component chosen is a random variable, C, with

$$P(C = c) = S(c) \tag{2}$$

for some measure $S(\cdot)$ over $\boldsymbol{C}$.

involves independent (i.e. from team to team) random selection by the different teams of the different components of the software development process from among the many competing candidates - e.g. formal specification and implementation languages, testing techniques, etc.

There is similarly a population of environments

$$\boldsymbol{E} = \{e_1, e_2 \ldots\} \tag{3}$$

from which one will be chosen in which the component will be required to operate. This environment is, in turn, a random variable E with

$$P(E = e) = Q(e) \tag{4}$$

for some measure $Q(\cdot)$ over $\boldsymbol{E}$.

For a particular component in a particular environment, there is a probability of failure

$$f(c,e) = P(\text{component } c \text{ fails in environment } e). \tag{5}$$

This can be thought of as the probability that this component will fail upon demand in this environment. It allows the possibility of a particular component to behave differently in successive *similar* demands, something which does not appear to be included in the original formulation of the Hughes model. Notice that this probability contrasts with the E&L equivalent, where it is replaced by a simple $(0,1)$ indicator function since a particular program version will *always* fail (or always succeed) on a particular input. For software it is often said that faults are 'systematic', meaning that program failure behaviour will always be the same in the same circumstances.

We are now interested in measures of failure behaviour such as

$$\pi(e) \equiv P(\mathit{fail} \mid E = e) = \sum_{C} f(c,e)S(c) = E_S(f(C,e)) \tag{6}$$

which is the probability of failure in a particular environment e for a randomly chosen component. It is the variation of this probability over the different environments, representing informally the differing stressfulness of the environments, that underlies the Hughes model. For a randomly chosen environment, E, $\pi(E)$ is also a random variable, with a distribution

$$F(\pi) = P(\pi(E) \le \pi). \tag{7}$$

This is equivalent to the distribution given in section 3 of [Hughes 1987]. However, our treatment here is more general than that represented in Hughes, and admits of the following dual formulation.

$$\phi(c) = \sum_{E} f(c,e)Q(e) = E_Q(f(c,E)) \tag{8}$$

is the probability of failure of a *particular component*, c, in a randomly chosen environment. So $1-\phi(c)$ is the reliability of component c. Different components have different reliabilities in this generalisation, in contrast to the Hughes model where this possibility is not considered. For a randomly chosen component, C, $\phi(C)$ is also a random variable with distribution

$$H(\phi) = P(\phi(C) \le \phi). \tag{9}$$

A particularly important measure is now the probability that a randomly chosen component fails in a randomly chosen environment, which is given by

$$E(\pi(E)) = E(\phi(C)) = E_{S,Q}(f(C,E)) = \sum_{C}\sum_{E} f(c,e)S(c)Q(e). \tag{10}$$

In the present notation, the Hughes results primarily concern the distribution of $\pi(E)$ over the different environments, and in particular the effect that the variance of this distribution has on the probabilities of simultaneous failure of several components operating in the same environment. Consider the simple case of two components selected randomly and independently to operate as a 1-out-of-2 system, i.e. the system works successfully if at least one component works. The probability of the system failing in a randomly selected environment is

$$\sum_{E}\sum_{C}\sum_{C} f(c_1,e)f(c_2,e)S(c_1)S(c_2)Q(e) = \sum_{E}\left(\sum_{C} f(c,e)S(c)\right)^2 Q(e)$$

$$= \sum_{E}(\pi(e))^2 Q(e) = E\left((\pi(E))^2\right) \tag{11}$$

which is bigger than the result we would obtain under the (incorrect) assumption that the two versions fail *independently* with the probability (10), since

$$E\left((\pi(E))^2\right) = \left(E(\pi(E))\right)^2 + \mathrm{Var}(\pi(E)) \geq \left(E(\pi(E))\right)^2. \tag{12}$$

Putting this another way:

$$P(\text{second component fails}|\text{first component failed}) = \frac{P(\text{both failed})}{P(\text{first component failed})}$$

$$= \frac{\left(E(\pi(E))\right)^2 + \mathrm{Var}(\pi(E))}{E(\pi(E))} = E(\pi(E)) + \frac{\mathrm{Var}(\pi(E))}{E(\pi(E))}$$

$$= P(\text{second component fails}) + \frac{\mathrm{Var}(\pi(E))}{E(\pi(E))}$$

$$\geq P(\text{second component fails}). \tag{13}$$

That is, knowing that the first component has failed in this randomly chosen environment *increases* our probability that the second component will fail. Informally, having observed the failure of one of the components makes us more confident that this is a 'stressful' environment, i.e. that the probability of failure is greater for *every* component, than would otherwise be the case. This can be formalised as follows

$$\pi(E)|\text{first component has failed} \overset{st}{\geq} \pi(E), \tag{14}$$

i.e. the conditional and unconditional random variables are stochastically ordered (see [Littlewood and Miller 1989] for proof). There is equality in (13) and (14) if and only if the variance is zero, i.e. all the probability in the distribution (7) is concentrated at a single point and there is no variation in stressfulness over the different environments. It

is only in this case that conditional independence of failures of two components results in unconditional independence, and we can resort to the more naïve model. In all other cases - and, it presumably follows, in all reasonable real-life situations - we shall have worse failure behaviour of the system than the naïve model suggests. Exactly *how much* worse depends upon the magnitude of the variance of the distribution of failure probabilities, as is shown in (12).

There are similar dual results to the above, concerning a randomly selected *single* component called upon to operate in *two* randomly selected environments, if we allow perfect restoration following failure:

$$P(C \text{ fails in } E_1,\ E_2) = E\left((\phi(C))^2\right) \geq \left(E(\phi(C))\right)^2 \tag{15}$$

which is the dual of (11), (12);

$$\phi(C)|\text{failure in first environment} \overset{\text{st}}{\geq} \phi(C) \tag{16}$$

which is the dual of (14). The reasoning here is similar to before: the fact that this component has failed in the first environment makes us think that 'it is probably a weak component', and thus will be more likely than otherwise to fail when presented with another environment. Proofs of (14), (16) are similar to those in [Littlewood and Miller 1989] for the software design diversity context. Notice that the dual results only really make sense in the hardware context if it is possible for the component to be restored by repair to the same state it was in prior to failure.

These results show the crucial importance that variability plays in several areas, and how misleading it can be merely to average out this variation when studying common mode failures. Most importantly, the average behaviour of an average component, (10), is not sufficient for us to know the reliability of a redundant 1-out-of-2 system, (11). In fact, of course, the general thrust of these arguments about the optimism of incorrect assumptions about failure independence apply much more widely than this simple 1-out-of-2 system. Hughes shows, for example, that in the 1-out-of-*n* case the failure process will be dominated by failures that are common cause in origin.

4 Forced diversity

4.1 A new model

We consider now a new model for forced diversity of hardware components, similar to the E&L model [Littlewood and Miller 1989] for software design diversity. The idea here is that we may have available two, or more, different types of components, *A*, *B*, . . . Components of type *A* may differ from those of type *B* because they represent two different designs, but all of the *A*s will have the same design, as will all of the *B*s. Alternatively, manufacturing processes may be known to be different, even though the designs are nominally identical: an example would be that different manufacturers used different testing regimes, so that the components weeded out during testing had succumbed to different *kinds* of stress (leaving components that were more resistant to that kind of stress).

This kind of diversity introduces another form of variation over and above that already discussed. In particular, in addition to variation among the *A* component reliabilities, and among the *B* component reliabilities, over the different environments,

there will be variation between *A*s and *B*s. We can capture this in an enhanced version of the notation used in the previous section. Keeping, for simplicity, to the situation of a 1-out-of-2 system, constructed from an *A* and a *B*, we have:

For the population of *A* components

$$\boldsymbol{C_A} = \{c_{A1}, c_{A2}, \ldots\} \tag{17}$$

the component chosen is a random variable, C_A, with

$$P(C_A = c_A) = S_A(c_A) \tag{18}$$

and similarly for *B* components. Selection of the environment is, as before, determined by (3), (4).

Within the population of type-*A* components things are as described above, with $\pi_A(e)$ representing the probability that a randomly chosen type-*A* component will fail in a particular environment *e*, similarly for type-*B*. Just as, in the univariate case of a single population of components, interest centres on the distribution of the probability of failure over different environments, so here we have the bivariate distribution

$$F(\pi_A, \pi_B) = P(\pi_A(E) \le \pi_A, \pi_B(E) \le \pi_B) \tag{19}$$

Consider now the case where a 1-out-of-2 system is built using components selected independently and randomly from the two populations. That is

$$P(C_A = c_A,\ C_B = c_B) = P(C_A = c_A)P(C_B = c_B) = S_A(c_A)S_B(c_B) \tag{20}$$

The probability that, for a particular environment *e*, the system fails is just

$$\pi_A(e)\pi_B(e) \tag{21}$$

representing the *conditional independence* of failures. But for a randomly chosen environment the *unconditional* probability that the system fails is, in an obvious notation

$$\sum_{\boldsymbol{E}} \sum_{\boldsymbol{C_A}} \sum_{\boldsymbol{C_B}} f(c_A, e) f(c_B, e) S_A(c_A) S_B(c_B) Q(e)$$

$$= \sum_{\boldsymbol{E}} \left(\sum_{\boldsymbol{C_A}} f(c_A, e) S_A(c_A) \right) \left(\sum_{\boldsymbol{C_B}} f(c_B, e) S_B(c_B) \right) Q(e)$$

$$= \sum_{\boldsymbol{E}} \pi_A(e)\pi_B(e)Q(e) = E\left(\pi_A(E)\pi_B(E)\right)$$

$$= \mathrm{Cov}\left(\pi_A(E)\pi_B(E)\right) + E\left(\pi_A(E)\right)E\left(\pi_B(E)\right) \tag{22}$$

$$= \mathrm{Cov}\left(\pi_A(E)\pi_B(E)\right) + P(A \text{ component fails})P(B \text{ component fails}) \tag{23}$$

$$> P(A \text{ component fails})P(B \text{ component fails})$$

if and only if $\mathrm{Cov}\left(\pi_A(E)\pi_B(E)\right) > 0$.

Putting this another way,

$$P(B \text{ component fails} | A \text{ component fails}) = \frac{\text{Cov}\left(\pi_A(E)\pi_B(E)\right)}{E\left(\pi_A(E)\right)} + E\left(\pi_B(E)\right)$$

$$> E\left(\pi_B(E)\right) = P(B \text{ component fails}) \textit{ if and only if } \text{Cov}\left(\pi_A(E)\pi_B(E)\right) > 0.$$

4.2 Discussion: some general results

The point about these results is that everything now depends upon the value of $\text{Cov}\left(\pi_A(E)\pi_B(E)\right)$, particularly its sign. There is a sense in which we can say that, all other things being equal, forcing diversity in this way guarantees that we shall do better than we would by selecting the pair of components randomly from a single population. In the event that the covariance term is negative, we even do better than we would under the assumption of *independence*. Even if, as seems likely, the covariance is not negative, the Hughes model is still *the worst case scenario.* Then $\pi_A(e) \equiv \pi_B(e)$ for all *e*. That is, the bivariate distribution $F(\pi_A, \pi_B)$ is concentrated on the line of unit slope in the (π_A, π_B) plane.

Just as the E&L model is essentially identical to the Hughes model in the case of a single population, so the model presented here is mathematically similar to that of [Littlewood and Miller 1989]. The only difference lies, as before, in the fact that a *program* fails 'systematically' in the sense that a particular program failing on a certain input will *always* fail on that input. In our case, in contrast, the component that fails in a particular environment today may not do so tomorrow. The mathematical implication of this is that the indicator variables of E&L and L&M are replaced by the probabilities $f(c,e)$, representing the probability that component *c* will fail in environment *e*. This does not affect the structure of the mathematical arguments significantly, and the results presented in L&M [Littlewood and Miller 1989] go through quite simply.

For example, it is possible to give quite general advice as to which of several different systems should be preferred among the many alternatives that could be built. Consider again a 1-out-of-*n* system. Clearly the reliability of such a system will depend not only upon the degree of dependence between the failure behaviours of the components, as discussed above, but also upon the individual component reliabilities. To make some *general* statements about the efficacy of this kind of forced diversity we can make some simplifying assumptions about indifference between types of components. We shall say we are *indifferent* between types *A* and *B* if we believe that permutations of the component type do not change our probabilities of failure. Thus for systems comprising *single* components, we shall be indifferent between *A* and *B* if

$$P(A \text{ system fails}) = P(B \text{ system fails}).$$

Similarly, we are indifferent between 1-out-of-2 homogeneous *A* systems, and similar *B* systems if, in an obvious notation

$$P(AA \text{ system fails}) = P(BB \text{ system fails}).$$

We then obtain the following result: *If we know that* A *components and* B *components are different, but we are indifferent between randomly chosen* AA *and* BB *systems, then we should build instead a randomly chosen* AB *system*.

In other words, a randomly chosen *mixed* system will be more reliable than either of the possible randomly chosen homogeneous systems. The result generalises to the case of 1-out-of-*n* systems if we have *n* types of components: it is best to select the components randomly one from each of the different types.

In the case where we wish to build a 1-out-of-*n* system, but have fewer than *n* different types of components, it can be shown that: *The best design is the one that uses all the available types of components, but uses each as little as possible*. Thus for example, an *AABBC* system will be preferable than an *AAABC* (see [Littlewood and Miller 1989] for details).

It should be emphasised that all these results are based upon averages, and concern what we should regard as the best system design *when we know nothing about the reliabilities of particular components in particular environments*. If we have more information, things can change dramatically. Thus, for example, for a particular environment *e*, we may know that type-*A* components are more reliable than other types, in which case if we have to build a 1-out-of-2 system we may prefer a randomly chosen pair of components to make an *AA*, than to make an *AB*. In fact, in L&M it is shown that for a particular *e* the best of *AA* and *BB* will be better than the average *AB*. In general, it can be shown that preferences that we have 'on average' are precisely reversed when we have sufficient knowledge to be able to talk of 'best of'. Thus, in the example of $n = 5$ above, the *best AAABC* will be more reliable than the *best AABBC*[5]

However, in spite of these qualifications, the results here are quite strong. They essentially say that, in the event of not having detailed information about the way that the reliabilities of the different types of components vary over different environments, and subject to certain indifference assumptions, 'the maximum application of diversity is the best strategy'.

In practice, we may have available sufficient information to undermine the indifference assumptions here. For example, we may know that, even on average, type-*A* components are more reliable than type-*B* components. In such a case the reliability of, say, a 1-out-of-2 system is determined by the trade-off between component reliability and component failure dependence: see (22). The information we need is contained in the joint distribution $F(\pi_A, \pi_B)$; in particular, we need the covariance and the means of the marginal distributions.

4.3 Calculation of system reliability: the use of failure data

We have seen that there are some general things we can prove about the advantages of using forced diversity rather than merely redundancy as in the Hughes model. In practice, though, we shall wish to have estimates of the actual reliability of systems we build in this way. A major advantage of the approach of Hughes, and its generalisation described here, is that everything depends upon distributions such as that for (π_A, π_B) given by (19). We can estimate these distributions from the data that we have collected over the operational behaviour of different earlier systems containing *A* and *B* components; in particular, there is no necessity for the systems that provide this information to be similar to the ones about which we wish to make predictions. All that

---

[5] The actual labels here are not significant: by *AAABC* here we simply mean a system built of three types of component, with three of the components being of the same type.

is needed is that we have information that ties together the failure behaviour, component type, and environment. It is worth emphasising that such information is much more likely to be available in quantity than will be the case for design diversity in software.

Let us assume that there are m environments in total. For environment e_j let us assume that we have observed n_{Aj} demands upon type-A components, of which r_{Aj} resulted in failure, and n_{Bj} demands upon type-B components of which r_{Bj} resulted in failure.

Further, let us assume that the mechanism for selecting environments is such that there is a probability p_j of selecting environment e_j, and that these probabilities remain the same for all selections of environment. Assume that we have observed that e_j has been selected q_j times ($j = 1,2, \ldots, m$).

A crude way to proceed now is to use the data to estimate the (conditional) probabilities of failure of the component types in the different environments, and then combine these with the estimates of the probabilities of selection of the environments to obtain an estimate of the unconditional distribution, (19), for the joint probability of failure of type-A and type-B components. A simple Bayesian argument for this case of Binomial trials [DeGroot 1970] gives an estimate

$$\frac{r_{Aj}+1}{n_{Aj}+2} \tag{24}$$

for $\pi_A(e_j)$, the probability of failure of a type-A component in environment e_j, with a similar expression for the failure of a type-B component. Note that these components fail (conditionally) independently in this environment. A similar argument, this time based upon the multinomial distribution, gives an estimate

$$\frac{q_j+1}{\sum_k q_k + m} \tag{25}$$

for p_j.

The unconditional probability distribution for (π_A, π_B) is then

$$P\left(\pi_A(E) = \frac{r_{Aj}+1}{n_{Aj}+2}, \pi_A(E) = \frac{r_{Bj}+1}{n_{Bj}+2}\right) = \frac{q_j+1}{\sum_k q_k + m} \tag{26}$$

which of course only has support on a finite number of points. Then, for example, the probability of failure of a 1-out-of-2 diverse AB-system will be

$$\sum_j \frac{r_{Aj}+1}{n_{Aj}+2}\cdot\frac{r_{Bj}+1}{n_{Bj}+2}\cdot\frac{q_j+1}{\sum_k q_k + m} \tag{27}$$

A naïve assumption of independence here would give a probability of system failure

$$\left(\sum_j \left(\frac{r_{Aj}+1}{n_{Aj}+2}\right)\frac{q_j+1}{\sum_k q_k+m}\right)\left(\sum_j \left(\frac{r_{Bj}+1}{n_{Bj}+2}\right)\frac{q_j+1}{\sum_k q_k+m}\right) \tag{28}$$

and, under the Hughes model, a 1-out-of-2 system comprising only type-*A* components would fail with probability

$$\sum_j \left(\frac{r_{Aj}+1}{n_{Aj}+2}\right)^2 \frac{q_j+1}{\sum_k q_k+m} \tag{29}$$

and similarly for a system made out of type-*B* components.

The point of all this, of course, is that it allows *in principle* for the probability of system failure given by (27) to be smaller than any of those given by (28) and (29). Naturally, such superiority cannot be guaranteed, but we are at least free from the tyranny of the Hughes model's *certainty* of positively associated failures. In practice, the extent to which we can gain extra confidence about a system from this approach will depend critically upon the nature of the correlation between the failure behaviour of the different types of components, given by the difference between the expressions in (27) and (28): all other things being equal, we wish this correlation to be 'as negative as possible'.

A less crude approach than the above, following that in Hughes' paper, would replace the point estimates of the probabilities of failure of the different types of component, (24), with the Bayesian posterior distributions from which these were derived. With uniform independent priors we have, conditionally, the probability density function

$$f(\pi_A,\pi_B|e_j) = C.{\pi_A}^{r_{Aj}}(1-\pi_A)^{n_{Aj}-r_{Aj}}.{\pi_B}^{r_{Bj}}(1-\pi_B)^{n_{Bj}-r_{Bj}} \tag{30}$$

where C is a normalising constant. Then the unconditional joint distribution of the failure probabilities has probability density function

$$f(\pi_A,\pi_B) = \sum_j f(\pi_A,\pi_B|e_j)\frac{q_j+1}{\sum_k q_k+m} \tag{31}$$

Then, for example, the probability of failure of a 1-out-of-2 *AB* system is

$$E\left(\pi_A(E)\pi_B(E)\right) = \int_0^1\int_0^1 \pi_A\pi_B f(\pi_A,\pi_B)d\pi_A d\pi_B \tag{32}$$

with similar expressions for different architectures. These results easily and naturally extend to more than two types of components. Thus the probability of failure of a 2-out-of-3 *ABC* system is

$$\begin{aligned} &E\left(\pi_A(E)\pi_B(E)\right) + E\left(\pi_A(E)\pi_C(E)\right) + E\left(\pi_B(E)\pi_C(E)\right) \\ &-2E\left(\pi_A(E)\pi_B(E)\pi_C(E)\right) \end{aligned} \tag{33}$$

All the terms in expressions such as (32) and (33) essentially involve moments of Beta distributions and are thus easily computed.

An advantage of these ways of computing system reliabilities is that they factor out the two different kinds of evidence needed - about the component behaviour in particular environments, and about the mechanism that selects environments. In the above it has been assumed that prediction of system reliability will take place under the same mechanism for selection of environments as the component failure behaviour was collected. But this is not necessary. If it is required to predict for a novel kind of environment selection - i.e. a system operating in circumstances that differ from those previously seen - it may sometimes be possible to replace (25) by p_js calculated directly, for example by direct observation in the new context.

4.4 An example

In his paper, Hughes considers some hypothetical data on a single system with four components, upon which a total of 20 demands have been made. This data is shown in Table 1.

Number of system demands	Number of component failures
16	0
2	1
1	2
1	3
0	4

Table 1: Hypothetical data for a 4-fold redundant system showing the number of system demands with a given number of component failures.

Let us now suppose that in fact the components from which the system is built are of two types, *A* and *B*, and that there are two components of each type comprising a system. Upon inspection of the above failure data, it is found that the failures corresponded to the different types of components as shown in Table 2.

Number of system demands	Number of *A* component failures	Number of *B* component failures
16	0	0
1	1	0
1	0	1
1	2	0

1	1	2

Table 2: The data of Table 1 when we have additional information that allows the type of component to be identified.

We must now identify the different 'environments' associated with the demands. In reality, such classification of demands into different environments will be carried out using engineering information about the nature of the demands. Here we shall assume, in the same spirit as Hughes, that all demands that produce the same pair of numbers for *A* and *B* component failures correspond to a single environment: *this is, of course, unrealistic and we adopt the assumption purely for illustrative purposes* Thus above we saw 16 demands from the environment corresponding to (0,0); 1 demand corresponding to the environment for (0,1); 1 demand for the environment corresponding to (1,0); 1 demand for the environment corresponding to (2,0); 1 demand for the environment corresponding to (1,2). There are 5 environments in total: we shall assume for simplicity that the possible remaining environments, corresponding to (0,2), (1,1), (2,1) and (2,2), do not exist.

Now line j of Table 2 corresponds to environment e_j. Then, in the notation of the previous subsection: m=5; n_{A1}=n_{B1}=32, r_{A1}=r_{B1}=0; n_{A2}=n_{B2}=2, r_{A2}=1, r_{B2}=0; n_{A3}=n_{B3}=2, r_{A3}=0, r_{B3}=1; n_{A4}=n_{B4}=2, r_{A4}=2, r_{B4}=0; n_{A5}=n_{B5}=2, r_{A5}=1, r_{B5}=2; q_i=16, q_2=q_3=q_4=q_5=1.

Substituting in (27) we find that the estimated probability of failure of a 1-out-of-2 diverse *AB*-system is 0.0656. This compares with 0.0906 and 0.0756 obtained from substitution in (29) for the estimated probabilities of failure of, respectively, *AA*- and *BB*- systems under the Hughes model. Thus the forced diverse system is preferable to each of the two possible homogeneous systems. However, in this case there is still positive covariance between the two different system types. This can be seen by substituting in (28), which gives a probability of failure of 0.0288 (the product of 0.18 and 0.16, being respectively the estimates of the probability of failure of single *A* and *B* components) under the naïve (incorrect) assumption of independence.

5 Discussion and conclusions

The main strength of all this modelling work is that it provides us with a formal conceptual framework within which we can reason probabilistically about dependencies in the failure processes of redundant and diverse systems. In the hardware context, the Hughes model played the same rôle as the Eckhardt and Lee model played for software: it provided a mathematical reason for the common-sense assumption that components do not fail completely independently in practice. The new model presented here provides an opportunity to avoid the worst implications of the Hughes model - it even allows *in principle* for failure behaviour *better than* independence.

In practice, of course, such 'negatively correlated' failure behaviour may be very rare. However, under the conditions of this new model, at least there is a sense in which the Hughes model can be regarded as the worst case.

In real applications of this work, it will be necessary to estimate what has actually been achieved, rather than to rely upon the more general results presented here. It is here that hardware engineers have a considerable advantage over their software

counterparts. As long as there is sufficient failure data, from previous use of the different component types, that 'covers' the environments within which the new system will operate, the distributions that are needed to compute system reliability will be estimable. This contrasts with the software diversity situation, where estimability is severely constrained by the practical limitations to the number of versions that can be developed. Thus the E&L and L&M models remain largely conceptual, whilst the Hughes model and the model presented here could be of practical assistance in computing the reliability of real systems.

The rôle played by variation in all this modelling is somewhat mysterious and surprising. In the Hughes model (and in E&L), variation of 'difficulty' or 'stressfulness' over the different environments is what prevents us achieving the ideal of independent failure behaviour that would, in turn, allow the achievement of arbitrarily high reliability by the use of sufficient redundancy. Thus here variation is *a bad thing*: the more variation we have here, the worse things are. In the new model, in contrast, the variation between the failure behaviours of different types of component is *a good thing*, and allows *in extremis* the achievement of component failure behaviour better than independent.

References

[Bourne, Edwards et al. 1981] A.J. Bourne, G.T. Edwards, D.M. Hunns, D.R. Poulter and I.A. Watson. *Defences against common-mode failures in redundancy systems,* UKAEA Safety and Reliability Directorate, 1981.

[DeGroot 1970] M.H. DeGroot. *Optimal Statistical Decisions,* New York, McGraw-Hill, 1970.

[Eckhardt, Caglayan et al. 1991] D.E. Eckhardt, A.K. Caglayan, J.C. Knight, L.D. Lee, D.F. McAllister, M.A. Vouk and J.P.J. Kelly, "An experimental evaluation of software redundancy as a strategy for improving reliability," *IEEE Trans Software Eng*, vol. 17, no. 7, pp.692-702, 1991.

[Eckhardt and Lee 1985] D.E. Eckhardt and L.D. Lee, "A Theoretical Basis of Multiversion Software Subject to Coincident Errors," *IEEE Trans. on Software Engineering*, vol. 11, pp.1511-1517, 1985.

[Hughes 1987] R.P. Hughes, "A new approach to common cause failure," *Reliability Engineering*, vol. 17, pp.211-236, 1987.

[Knight and Leveson 1986] J.C. Knight and N.G. Leveson, "Experimental evaluation of the assumption of independence in multiversion software," *IEEE Trans Software Engineering*, vol. 12, no. 1, pp.96-109, 1986.

[Littlewood and Miller 1989] B. Littlewood and D.R. Miller, "Conceptual Modelling of Coincident Failures in Multi-Version Software," *IEEE Trans on Software Engineering*, vol. 15, no. 12, pp.1596-1614, 1989.