Design Diversity: an Update from Research on Reliability Modelling

Bev Littlewood, Peter Popov, Lorenzo Strigini

Centre for Software Reliability, City University

London, U.K.

Abstract

Diversity between redundant subsystems is, in various forms, a common design approach for improving system dependability. Its value in the case of software-based systems is still controversial. This paper gives an overview of reliability modelling work we carried out in recent projects on design diversity, presented in the context of previous knowledge and practice. These results provide additional insight for decisions in applying diversity and in assessing diverse-redundant systems. A general observation is that, just as diversity is a very general design approach, the models of diversity can help conceptual understanding of a range of different situations. We summarise results in the general modelling of common-mode failure, in inference from observed failure data, and in decision-making for diversity in development.

1 Introduction and Background

Diversity is a common design approach for protection against common-mode failures in redundant systems, mostly used in critical applications. It is hoped that if redundant channels are implemented in different ways (diverse "versions"), the risk of common design flaws causing common failures will be reduced. The growing adoption of software-based systems, and the attendant doubts about the risk caused by design faults in the software, justify increased interest in diversity. Well known examples of diversity in software are in the aerospace and railway industries, but some form of diversity is present in many software systems.

Research in software diversity was very active in the 1980s, including several experiments on forms of "multiple-version programming" and reliability modelling work. Activity then subsided for about 10 years. In the last few years, the Centre for Software Reliability (CSR) at City University has done some novel work in modelling the reliability of diverse systems to gain insight for supporting decisions. This paper is meant as a short summary of these results to date; mathematical details are available in several published papers. For a summary of previous results and references to previous literature, we refer the reader to [Littlewood 2001; Littlewood 1996].

There are two kinds of open technical questions of practical interest:

- ***achievement of reliability:*** a manager or designer wishing to apply diversity has no well-founded guidance as to which methods will be most effective or cost effective, among the many that are available. Diversity is sought by decisions in project management and system design: examples include separating the development processes, dictating different algorithms, different programming or specification languages, etc. But the effects of these decisions on failure diversity (i.e., on reducing the correlation between failures of the two channels) is indirect, as shown summarily in Figure 1 below. Very little is known about how to choose the "diversity-seeking decisions" shown at the top so as to produce effectively and efficiently the diversity in failure behaviour, shown at the bottom, and thus improved system reliability and safety.

Originally published in Proc of the 9th Safety-critical Systems Symposium, Bristol 2001
©Springer-Verlag London Ltd

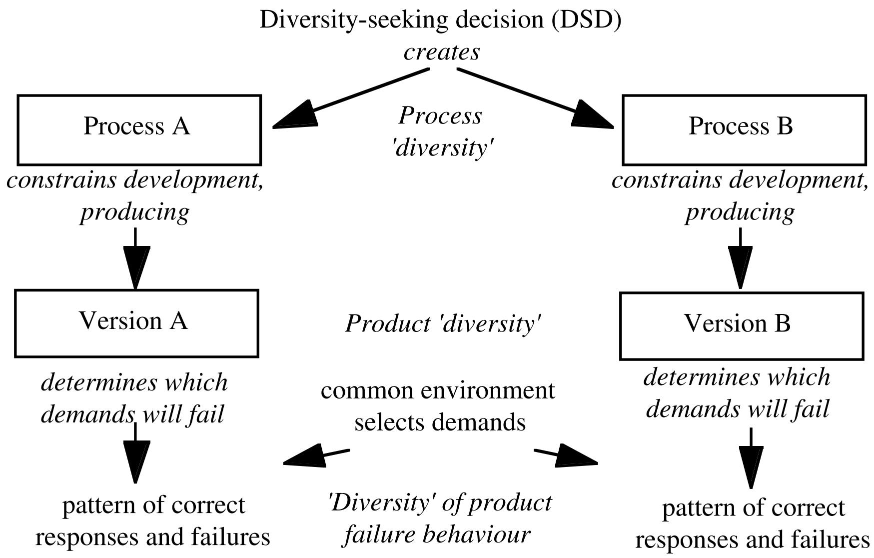


Figure 1. The different facets of "diversity" and their inter-dependence, for a two-version system.

- ***assessment of reliability:*** when it comes to assessing the reliability of a diverse-redundant system, we would like to exploit the knowledge that diversity is present, both to claim higher system reliability and if possible to quantify it. Theoretical and practical research in the 1980s showed that one cannot assume failure independence even if the two channels are developed in a "strictly independent" fashion. Conceptual models developed by Eckhardt and Lee (*EL model*), and Littlewood and Miller (*LM model*) (and by Hughes for physical failures) suggest that if the two development processes are similar, though rigorously independent, positive failure correlation is to be expected. These models are based on the idea that for the builders of diverse versions of a program some demands will presumably be more difficult - more error-prone - than others. So, even if diverse versions are built "independently", their failures are more likely to happen on certain demands than on others, and this leads to positive correlation. If a project is managed so as to "force" diversity, independence or even negative correlation between failures of the two channels could be achieved. But there is no practical method yet of judging the level of correlation between the channel failures, and hence of system reliability, short of extensive testing of the system as a whole. That is, although diversity improves reliability, the knowledge that diversity is present brings no quantifiable advantage during assessment.

Reliability modelling may be useful for both problems. The problem of choosing how to apply diversity for improving reliability is one of predicting the likely outcomes of the methods chosen; that of assessment is a problem of predicting the future behaviour of a specific system, using any specific information available about it.

Given a redundant system, expressing its probability of failure in terms of those of failures of its components and combinations thereof is a well understood process. The open problem is in assigning a probability to the joint failure of two components, when failure independence cannot be assumed. This probability - how to evaluate it or reduce it - is the topic of our research. Our studies thus refer usually to the simplest diverse system, depicted in Fig. 2: a two-channel, 1-out-of-2, diverse, demand-based system, as could be for instance a protection system. We expect that better understanding of diversity in this basic case will lead to better understanding for more complex applications.

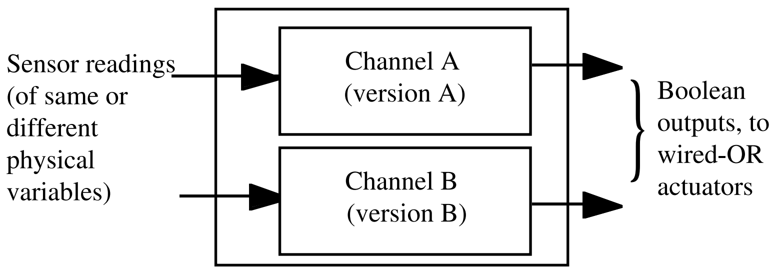


Figure 2. Reference system for this discussion. The system is subjected to discrete demands, and the measure of interest is the systems probability of failure per demand (*pfd*), i.e. the probability of both diverse channels failing on the same demand.

We have divided this summary into the following themes: what can be expected from diversity in general; design decisions for building diverse systems; assessment, and design for assessment, of diverse systems; diversity in the development process. In the last section we discuss future developments.

2 What Can Be Expected from Diversity in General

In the system of Fig. 2, diversity obviously improves dependability: the *pfd* of the system cannot be higher than that of either channel. For more complex configurations, which do not by themselves guarantee reliability improvement, all known evidence still points to some degree of improvement. However, the practical questions about the usefulness of diversity, for any specific application and industrial context, are:

- does it deliver substantially better reliability (or substantially better chances of achieving high reliability) than can be achieved without it? This would be attractive in applications where very high reliability is required;
- alternatively, does it deliver a reliability improvement more cost-effectively than alternative techniques?

2.1 "Independent Faults" Models

The EL and LM reliability models provide useful insight (see [Littlewood 2001], but have at least two limitations:

- They predict the *average* [un]reliability of versions and pairs of versions. To quantify the risk of exceeding a desired upper bound on failure probability, we need instead *distributions* of the *pfd*.
- They use parameters that would be unknown in practice when trying to predict the reliability of a system.

To improve on this, we modelled plausible assumptions on how faults are created in diverse software, and analysed their consequences. The paper [Popov 1999a] explains our method and some interesting conclusions about how development processes may affect reliability of diverse systems.

To overcome the limitations of the EL/LM models, we needed to add extra assumptions to theirs. These models use as parameters the probabilities of each demand being a failure-causing demand; they need to be extended with the *joint* probabilities of any two (or more) demands being failure-causing demands. We know from experience that faults affect *sets* of many possible demands. So, we posited the following simplified model of how faults are created in programs and how they cause failures:

- there are finite sets of *potential faults*, each potential fault causes a certain set of demands to become a failure region; the sets are the same for all versions;
- for each possible version, the presence or absence of each fault (which is a random event) is independent of the presence of any other fault, and of the presence of any fault in another version. This is the novel assumption that allows us to predict distributions of *pfd* of versions and systems.

Even with very simple versions of such models [Popov 1998a], interesting observations are possible. For example, one could observe the effect of diversity on reducing the "tails" of the distribution of the *pfd* - essentially, reducing the risk of unacceptably "bad" systems that, as far as we know from experience, may be present even with "good" production processes. A conjecture that this model suggests is that the efficacy of diversity is affected by the probabilities of individual potential faults, more than by their contributions to the *pfd* of the versions they affect.

We then studied [Popov 1999a] the more general model and what it predicts about the reliability gains given by diversity. An interesting conclusion is that an assessor who has estimated an upper bound on the *pfd* of a single version, and an estimate of the maximum likelihood of any one fault being present in the software, can then give a conservative bound on the *pfd* of a two-version system, given by the product of the two. Assessors and project planners often need to reason with such rough estimates of probabilities that they cannot measure (e.g., when judging achieved SILs) and this result can assist their judgement or help them to check it for consistency. Another question we ask in this paper is how process improvements would affect the advantages of diversity. It appears that the answer must depend on the specific kind of improvement, and we can actually show criteria for decision (see also section 2.3).

In conclusion, we expect these models, although necessarily somewhat unrealistic in their details, to offer these advantages:

- a more convincing analysis than previously available of the implications of various plausible assumptions about diverse developments;

- a better chance of empirical verification than is possible for the informal claims often made for (or against) diversity.

2.2 How Effective is Functional Diversity?

Functional diversity is commonly thought to be more effective than simple "design" diversity. "Functional diversity", in this context, means that the redundant channels differ in more than the internal implementation of their software: they also differ at least in the physical variables they sample and the actuators through which they affect the controlled system (and often in the physical principles on which their computations are based, the implementation technology and so on). Thus, to give a simplified example, a reactor protection system might comprise two versions, one of which makes its trip decision based on temperature inputs, and the other on pressure inputs.

Functional diversity is an attractive way of forcing the two design teams to be "intellectually diverse" in their solutions to the design problem. If the designs are "very different" in some meaningful way, there is a good chance that they will differ usefully in the faults they contain, and thus tend not to fail together. The practical intuition here is straightforward, and it thus seems a plausible approach to the *achievement* of reliability in the fault-tolerant system. The question we address in our work [Littlewood 1999] is *how much* could be claimed for the use of functional diversity: specifically, could a claim of *independence* of version failures be supported?

This claim could be stated as follows: if a thorough analysis of the design shows no direct source of common-mode failures (e.g., common software modules between the channels), could we then assume that any failures of the two channels are statistically independent? (any suspected sources of common-mode failures can then be taken into account by applying some correction factor or claim limit on the system reliability).

In terms of the model, we show that such claims for independence between functionally diverse systems seem rather unrealistic. Instead, it seems likely that functionally diverse systems will generally exhibit positively correlated failures, and thus will be less reliable than a simple assumption of statistical independence would suggest.

Our model of functional diversity generalises the earlier EL, LM and Hughes models. It is based again on the notion of variation of "difficulty" - in the example here within the "pressure" and "temperature" demand subspaces P and T - and in particular how these difficulty functions are correlated over the allowable set of demands in $P \otimes T$. Only if "difficulty" did not vary (for the builders of either channel) between demands could we expect independence to hold, without detailed knowledge about the difficulties for the two channels. One interpretation of our result is that functional diversity is just a more general kind of forced diversity, similar to that already seen for diverse software versions that execute *the same* inputs as one another.

Of course, this reasoning does not affect the argument that functional diversity is an effective way to pursue high reliability. What is not possible, however, is to claim that functional diversity is sufficient in itself to justify an assumption of independence in the version failures. It leaves the system assessor with the task of evaluating precisely *how* dependent the versions are before he/she can evaluate the reliability of the system. This is not easy, as we have seen in other contexts.

2.3 Is It Better to Use Diversity or to Seek High Reliability in a Single Version?

Whether diversity is a convenient means for delivering high reliability has long been a subject of debate. For instance, [Hatton 1997] strongly argues that design diversity is now more effective than other methods, and that its cost-effectiveness should increase with improvements in the reliability delivered by common development processes (a contention shared by other experts). He first considers that the reliability advantage given by diversity in the Knight-Leveson experiment [Knight 1985]. was arguably greater than what "state of the art" development processes usually give over "ordinary" processes. He then points out that, if versions failed independently, increasing the reliability of the versions would also increase the reliability gain given by diversity. Although no similar result is proven for the general case of non-independent failures, some experimental results point in this direction, and he concludes that the balance of evidence is in favour of diversity as a means for achieving high reliability.

This prompted us to analyse the issue from an alternative viewpoint, since we also believe that the potential of diversity is often dismissed without proper consideration. We conclude that there is insufficient basis for deciding whether improved processes (in the sense of general improvements in delivered reliability) will usually make diversity more or less effective a solution, and hence how it will affect a comparison with other ways of delivering improved reliability. Our argument is two-fold:

- in terms of modelling [Popov 2000] if we model the effects of process improvements in the most intuitive way - as a decrease in the probabilities of each residual fault being present in the finished versions - it is apparent that process improvement does not necessarily favour diverse systems; whether it does depends on the detailed values of the model parameters;

- in terms of empirical data, we examined [Littlewood 2000c], published data from two experiments which produced many versions of the same software [Knight 1985; Eckhardt 1991]. In the former experiment, the versions with higher reliability seemed also to produce the higher reliability gains when combined in fault-tolerant configurations; in the latter, usage profiles in which the versions were individually more reliable were also those in which combining them into fault-tolerant configurations would produce the lesser gains.

Decreasing gain with reliability growth of the version was also observed in a simulation study of fault-tolerant software subjected to debugging [Djambazov 1995].

If we cannot expect a trend towards higher gains from diversity as versions improve, it becomes more difficult to trust generalisations from the results of a few experiments, either to expect that diversity will generally beat other reliability improvement techniques or that it will become increasingly cost-effective.

All these are negative conclusions about the generality of claims that can be made for diversity (or against it, for that matter) at the current state of knowledge. It must be underscored that they are not negative conclusions about the effectiveness itself of diversity.

3 Design Decisions for Building Diverse Systems

Our modelling work mostly addresses simple, stylised models: they are meant either to support general insight or to be practically applicable without requiring the estimation of too many parameters; does this work offer any useful indications for decisions in software development, which involve multiple variables and complex interdependencies? While modelling results cannot produce any quick and universal recipe for building diverse systems, they offer, in our opinion, a useful viewpoint from which to consider the factors and the options ("diversity-seeking decisions") available in a specific project.

We take the viewpoint that imposing constraints on developers of two diverse versions is a way of introducing artificial differences between the development problems presented by two versions (see Fig 1). The hoped-for result is that the "hard problems" for the two development teams will differ and lead to any failures being unlikely to occur together in the two versions.

Many recommendations have been made about methods for forcing diversity (we call them "diversity-seeking decisions", or *DSDs* for brevity). Examples include: using different development environments, different tools and languages at every level of specification, design and coding, implementing each function with different algorithms, applying different V&V methods, etc.

The problem with these recommendations is that they do not usually address how (and thus why and whether) these DSDs are supposed to improve the reliability of the delivered system. Their main weakness is that any experience on which the recommendations are based is usually experience about the likelihood of *faults*, not of *failures*. Yet, we know that system reliability is determined by the relationships between the likelihoods of the faults being present and their contributions to unreliability. The probabilities of faults alone are not a sufficient criterion for decisions: a DSD, for instance, might minimise the probability of common faults for those potential faults that have negligible effect on failure probability, so that the DSD brings no practical gain. However, we acknowledge that *if we have little evidence of the relationship between faults and failures*, aiming at reducing common faults, via sound scientific use of the available knowledge, may be the best approach available.

The experimental approach used so far to estimate the advantages of software diversity has probably reached its limits. Experiments become less and less affordable as we try to support specific project decisions (as opposed to simply inquiring whether diversity may be generally useful). The space of the experimental control variables is huge and the risk that the results would differ between the laboratory and specific real-world projects can never be excluded. We do need to be able to generalise from the results of experiments, and this means that we need to test theories about the mechanisms through which the DSDs affect reliability (which will be present in all projects, acting in greater or lesser degree depending on the specific circumstances), rather than just about how much they affect it (which is bound to vary with these circumstances). In [Popov 1999c], we suggested ways of tackling some of the important questions by *affordable* experiments, for instance by investigating the intuitive assumption that *fault* diversity increases *failure* diversity.

We have also analysed in some detail the various plausible ways in which DSD may be effective: e.g., diversity in failure behaviour may arise from conceptual differences in the mistakes made by developers, but also from the different effects that conceptually similar mistakes have on system versions with different internal structures. Another effect of differences in internal structures may be that of producing different subdivisions into modules so that in-the-small design problems may be different for the two teams, even in presence of similar in-the-large specifications. In practice, it may be impossible to tell apart these various effects *a posteriori* (we can analyse them for individual faults and failures, but not for statistical distributions of faults and failures given a certain development process), but understanding them is necessary to make sense of the arguments for and against the various DSDs.

The report [Littlewood 2000e] reviews categories of DSDs and discusses their expected advantages in light of the general discussion of the mechanisms of action, and of the anecdotal evidence available in the literature about their perceived effects in different circumstances.

In choosing DSDs, an obvious criterion is to try and match them to the perceived threats against which diversity is being applied. It is often claimed that the main threat, in the presence of high-quality development processes, comes from the upstream phases of development, e.g. from faults in requirements. This suggests that diversity is most valuable in these same upstream phases, e.g. as functional diversity. However, some limits to this argument are worth pointing out. Whether a development process offers sufficient protection against "low-level", coding errors has to be assessed in the specific circumstances rather than assumed on the basis of very general evidence. Even with functional diversity, functional software building blocks in two channels may be substantially equivalent (e.g., implementing common mathematical functions) and thus in practice not affected by the intended "forcing" of diversity. Last, platform-level (hardware, operating systems) design faults in the diverse channels may be sensitive to similar plant conditions, e.g. causing overloads, so that diversity in the platforms will be necessary even with high-level diversity in applications.

4 Reliability Assessment and Design for Assessment

Although we have evidence that fault tolerance improves reliability, research results so far implied that it does not help to *assess* reliability. We can make claims for the "general efficacy" of diversity as a design approach - but such evidence would be weak for any *particular* system.

The EL and LM reliability models for diverse fault-tolerance gave results about what might be expected *on average*. These may be useful for general design guidance. But for accepting a certain finished system, it is necessary to predict its specific reliability (albeit with unavoidable uncertainty, represented via probability distributions or confidence intervals).

4.1 Modelling Reliability of a *Specific* Fault-Tolerant System

First, we need a model that represents the failures of a specific fault-tolerant system. We developed such a model [Popov 1998b], which is actually mathematically similar to the EL and LM models. A useful property of this model is in allowing some degree of prediction on system reliability from knowledge about the two versions. Given only limited knowledge - estimates of the versions' probabilities of failure, disaggregated by subdomains of their common demand space - we can calculate useful bounds on the probability of system failure on a random demand.

A claim limit for a fault-tolerant pair can be based on the plausible belief that failure independence between versions in each subdomain is an optimistic assumption: we would typically expect their failures to be positively correlated, as the EL model suggests. We also suggested [Popov 1999b] an upper (pessimistic) bound on system *pfd*, based on the obvious fact that, again for each subdomain of the demand space, the system is at least as reliable as the more reliable of the two versions (channels). This pessimistic bound on system *pfd* is practically useful in several cases: i) if it is close to the lower bound, they together define a good approximation to the "true" probability of system failure; ii) if the versions' reliabilities for different subdomains are negatively correlated (which is desirable), it may be lower than the (marginal) probability of failure of the better version, the only other known conservative bound. One should note that whether case ii holds depends on the specific data observed for a specific pair of versions: our conservative bound cannot be chosen *a priori* as the more convenient bound to use.

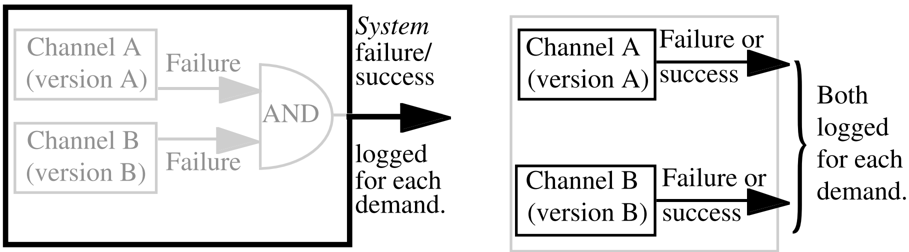
The reliability estimates for the two versions (for each subdomain) can typically be obtained from failure counts over periods of operation or realistic testing and will take the form of point estimates or of confidence bounds. Both forms can be used to calculate the proposed bounds. Typically, point estimates are misleading if few or no failures were observed. Useful confidence bounds, instead, can always be derived, even if no failure is observed. Uses of both kinds of estimates are illustrated in [Popov 1999b] using published data from an experiment on software fault-tolerance [Eckhardt 1991]. The data demonstrate the usefulness of this way of estimating reliability. For some usage profiles, our upper bounds for the system *pfd* were lower than the product of the estimates of the versions *pfd*s (over the whole demand space). That is, this more refined use of the available data would allow one to trust higher reliability than could be derived otherwise, even with an over-optimistic assumption of independence.

4.2 Bayesian Inference for Reliability Estimation of Fault-Tolerant Software

The method just described applies when we separately consider the failure records of the two versions. If we can observe the two-version system in operation (or realistic testing), we have additional, direct information about the *joint* failures of the two versions. Estimating system reliability from these data is easily done in a *black-box* fashion, counting just the number of *system* (i.e., common to the two versions) failures (Fig. 3a). However, this amounts to throwing away useful information: at least in testing, we can usually see any failures of either version, even when the presence of the other version masks them (avoids *system* failure). In [Littlewood 2000b] we examine this problem of inference from *clear-box* testing data (Fig 3b). For each demand, we observe a pair of binary random variables: version *A* fails (or not), version *B* fails (or not). The intuitive idea here is that by using this additional knowledge, rather than treating the system as a black box, we might be able to gain more confidence about its reliability. This is desirable, as black-box inference from an affordable amount of testing will often produce only modest reliability predictions [Littlewood 1993].

For example, if we were to observe many tests and observe a reasonable number of failures of each version *but no common failures*, we might reasonably claim that the failure dependence was low. Can we then claim a greater system reliability than could be claimed just from the evidence of (presence or absence of) common failures (the black box case)?

A Bayesian approach appears most suitable for this problem. In our work, we specify the correct inference procedure for this scenario, and show that black-box and clear-box inference will indeed produce different results.



3a. “Black box” scenario 3b. “Clear box” scenario

Fig. 3. Black-box vs. clear-box inference

Bayesian inference requires an assessor to start with *prior* distributions (a description of the assessor's beliefs before observing the failure data) for the model parameters - here, for the probabilities of failure of each version separately and of joint failure. We explored the problem of choosing forms of prior distributions to represent plausible beliefs for an expert assessor before he/she sees the failure data. The purpose here is both to simplify the task of assessors and (if possible) to reduce the computational cost of the procedure. This turns out to be a surprisingly difficult problem, and much of our paper is concerned with identifying pitfalls for the unwary. For example, if one applies the common method of choosing a "conjugate" prior - here, the *Dirichlet* distribution, corresponding to a *multinomial* likelihood function - the resulting claims that can be made about *system* reliability are exactly the same as the black-box results: the additional knowledge used produces no benefit.

We also demonstrate other simplified ways of building prior distributions. Some have advantages - e.g that of erring on the side of pessimism - in specific situations. However, none of these simplified solutions is useful in all situations.

In conclusion, we have confirmed that reliability predictions using all the "clear box" data about a fault-tolerant system may be different from those obtained from "black-box" data. So, this approach is worth using to check on the conclusions from other forms of assessment. On the other hand, the results about convenient ways of expressing the priors are rather tentative and are mainly warnings about the unreliability of intuition in this situation.

4.3 The Use of Proof in Diversity Arguments

We turn here to another way in which diversity can support the demonstration of high reliability.

It is common in several industries to have a form of diversity or redundancy in which the different channels or versions have different levels of trust placed in them. In some cases a highly functional primary system is backed up by a simple secondary procedure. For example, the present UK air traffic control system can revert to manual operation, involving paper records of aircraft movements, in the event of certain types of computer failure. Some computerised fly-by-wire aircraft flight control systems have a series of successively more degraded modes of operation, providing less and less functionality [Briere 1993]. In the case of the UK Sizewell B nuclear reactor, the protection system has two elements: a computerised Primary Protection System (PPS), and a hard-wired Secondary Protection System (SPS) [Hunns 1991].

All these examples have in common one or more processes providing high functionality at the price of complexity, backed up by simpler (but less functionally capable) processes. This design principle seems sensible, but its efficacy will clearly vary from one application to another.

The extensive extra functionality of the primary systems in these examples is present, of course, for good reason. The increasing use of software-based systems, in particular, seems to provide opportunities for novel functionality which, in addition to improving efficiency, sometimes increases safety and reliability in new ways. Thus the software-based Sizewell PPS is more complex than earlier non-computerised systems partly because it provides novel safety advantages (e.g. via the provision of extensive built-in hardware self-testing capabilities).

The problem, of course, is that a fairly complex software-based system must usually be assumed to contain residual design faults, which make its reliability or safety difficult to assess. What is needed in such cases is the ability to take advantage of the extra functionality that software can provide, whilst still being able to make appropriate safety claims. Architectures like that of the Sizewell safety system promise to do this: in [Littlewood 2000a] the problem of *assessing* the reliability of such systems is examined in a novel way.

The idea here is that a sufficiently simple secondary system has a *possibility* of being completely free of design faults. It may then be possible to claim that the probability of failure on demand of the overall system is simply the product of the *pfd* of the primary and *probability of incorrectness* of the secondary. If the first of these could be *measured* (e.g. by simulated realistic testing) at 10^{-3}, and the second estimated (e.g. on the basis of general experience) at 10^{-4}, a claim for the overall system *pfd* of 10^{-7} could be supported.

This is an "independence" argument, but *not* the discredited claim for independence between failures of two versions. Here the probabilities concern quite different things - failure on demand, and incorrectness. If the secondary were a software-based system, sufficiently simple to be open to formal verification by proof, it may be reasonable to claim that a failure in the proof process was independent of failures on demand of the primary version.

Notice that here a perfect secondary implies a perfect safety system - failures arising from software faults are impossible because the secondary always works correctly (with respect to design faults - physical faults are covered by other assessment procedures). Although *certain* perfection of the secondary is unlikely in real applications, the observation reveals how the architecture described here differs from a more conventional 1-out-of-2 system, where there is often a near symmetry of treatment of the two versions. If the protection function is intrinsically fairly simple, the secondary can be made simple, with the hope that it can be proven correct with low probability of error, and the primary can have extensive functionality, since its reliability requirement is sufficiently modest to be demonstrated by direct evaluation from testing.

5 Diversity in the Development Process

One thing that has struck us forcibly during our recent research has been the sheer ubiquity of diversity. In one sense this is not surprising: the notion of "two heads are better than one", or "don't put all your eggs in one basket" is age-old. On the other hand, the formal models of diversity were developed with quite narrow objectives - to understand and formalise diversity in software design - and yet they find application in much wider fields.

5.1 Application of Diversity in Software Fault Detection and Removal

The work described in [Littlewood 2000d] is concerned with diversity in software development process - specifically, in procedures for finding and removing software faults in the development of a *single* program.

We show that the theory of "difficulty functions", previously applied in design diversity, applies here as well. Here, "difficulty" represents the difficulty of finding a fault. For example, the reliability gains from repeated applications of a particular fault finding procedure are not statistically independent (there is a law of diminishing returns) - such an incorrect assumption of independence will always give over-optimistic results. When we have *diverse* fault finding procedures, however, things are different: here it is possible for effectiveness to be even greater than it would be under an assumption of statistical independence. We found theorems which show that, as in design diversity, diversity of fault finding procedures is "a good thing", and should be applied as widely as possible.

Most work in the software engineering literature on the efficacy of fault finding procedures is about assessing and comparing their *individual* efficacies. This is important, but in practice several of these techniques will be employed together. There are some well-known intuitions about such combinations of procedures: we all know, for example, that it is best to use procedures that are effective in some general way; but we equally know that any single such procedure may miss a whole class of faults even when applied most extensively. Even when we know that procedure *A* is better at fault finding than *B*, we would be wary of using *only A* because it may have little chance of finding certain faults that *B* instead finds quite easily. Our work formalises intuitions of this kind.

The key to understanding how best to apply different fault finding procedures lies in understanding the interplay between, on the one hand, the efficacies of the individual procedures (in single and multiple applications), and on the other hand the dependence between their "difficulty functions". These represent the way in which fault finding procedures vary in their effectiveness from one fault to another (and from one procedure to another).

Compared with its mathematically similar equivalent in design diversity, this model appears strikingly easier to apply to practical decisions. In an experiment based on a railway signalling application of diverse fault-finding, we were able to obtain estimates of the parameters representing procedure effectiveness and diversity, and some initial evidence that the model could be used for decisions in real projects.

Intuitive notions of diversity in fault finding have been around for a long time, and are used informally quite extensively, but they have lacked a rigorous formal basis. In particular, it has not been clear what were the important factors to *measure*. This work is the start of such a formal measurement-based understanding. We hope that it will lead to a theory of fault removal that allocates different fault finding procedures optimally, taking account of the likely distribution of fault types.

We expect to extend this work also to shed light on the role of diversity in the various phases of development of diverse versions in a fault-tolerant system.

6 Discussion and Future Work

The modelling work we have described has produced some encouraging progress. It is remarkable that the basic concept of 'variation of difficulty' seems suitable for formalising and understanding so many different problems.

However, we do not hide that many of these results are only useful for better understanding these complex, counter-intuitive problems: they do not lead to simple, general recipes for design and assessment. Such understanding is necessary before it is possible to begin *engineering* diverse fault-tolerant systems with dependability assurance founded on formal models. An important advantage of the formal models we have developed is that they provide a precise language for the discussion of some difficult issues: consider, for example, the rather loose way in which words like "independent" and "diverse" are sometimes used in discussions about fault tolerance.

Difficulties remain in using some of this theory - most particularly in populating the models with estimates of their key parameters when dealing with real systems - but we are now in a position to give advice to engineers on some substantive issues. We can advise on pitfalls in areas where intuition seems to be sometimes very unreliable. Not all this news is good: e.g. the demonstration that functional diversity does not, of itself, guarantee the validity of claims for failure independence.

We are now extending this work in various directions, in a set of recently started projects: DISPO-2 (DIverse Software PrOject-2), DOTS (Diversity with Off-The-Shelf components, in collaboration with the University of Newcastle-upon-Tyne), and the multi-university collaboration DIRC (Interdisciplinary Research Collaboration in Dependability of Computer-Based Systems). One of the extensions concerns the application of current models to practical case studies. Another one is their extension to encompass multiple factors of diversity in development. We will perform experiments to test some of the conjectures described in section 3. Other important extensions concern the area of application of diversity and of these modelling methods. For instance, when building systems out of off-the-shelf software 'of uncertain pedigree', architectures with diverse redundancy can be a cost-effective means for modest gains in dependability and assurance: they can be built without extensive knowledge of the internals of the off-the-shelf parts, while taking advantages of their low cost. Last, a form of diversity in *reasoning* is commonly used in the area of assessment, by developing two or more complete, 'independent' arguments to support a critical decision like that of accepting a certain critical system. We will attempt to formalise and clarify how and to what extent these practices produce additional assurance.

Acknowledgments

This paper is derived from the final report for project DISPO (DIverse Software PrOject), funded by Scottish Nuclear (later British Energy) at City University and Bristol University. The work described was mostly funded by project DISPO and by project DISCS (Diversity In Safety Critical Software), funded by the Engineering and Physical Sciences Research Council at City University and at the University of Newcastle-upon-Tyne.

References

Note: most papers by the present authors are available at URL: http://www.csr.city.ac.uk/projects/diversity

[Briere 1993] Briere D., Traverse P.: Airbus A320/A330/A340 Electrical Flight Controls - A Family Of Fault-Tolerant Systems. Proc. 23rd International Symposium on Fault-Tolerant Computing (FTCS-23): 616-623, Toulouse, France 1993.

[Djambazov 1995] Djambazov K.B., Popov P.: The effects of testing on the reliability of single version and 1-out-of-2 software. Proc. 6th Int. Symposium on Software Reliability Engineering, ISSRE'95: 219-228, Toulouse 1995.

[Eckhardt 1991] Eckhardt D.E., Caglayan A.K., Knight J.C., Lee L.D., McAllister D.F., Vouk M.A., Kelly J.P.J.: An experimental evaluation of software redundancy as a strategy for improving reliability. IEEE Transactions on Software Engineering, 17: 692-702, 1991.

[Hatton 1997] Hatton L.: N-Version Design Versus One Good Version. IEEE Software, 14: 71-76, 1997.

[Hunns 1991] Hunns D.M., Wainwright N.: Software-based protection for Sizewell B: the regulator's perspective. Nuclear Engineering International, September: 38-40, 1991.

[Knight 1985] Knight J.C., Leveson N.G., Jean L.D.S.: A Large Scale Experiment in N-Version Programming. Proc. 15th Int. Symp. on Fault Tolerant Computing (FTCS-15): 135-139, Ann Arbor, Michigan, USA 1985.

[Littlewood 1993] Littlewood B., Strigini L.: Validation of Ultra-High Dependability for Software-based Systems. Communications of the ACM, 36: 69-80, 1993.

[Littlewood 1996] Littlewood B.: The impact of diversity upon common mode failures. Reliability Engineering and System Safety, 51: 101-113, 1996.

[Littlewood 1999] Littlewood B., Popov P., Strigini L.: A note on reliability estimation of functionally diverse systems. Reliability Engineering and System Safety, 66: 93-95, 1999.

[Littlewood 2000a] Littlewood B.: The use of proof in diversity arguments. IEEE Transactions on Software Engineering: to appear, 2000.

[Littlewood 2000b] Littlewood B., Popov P., Strigini L.: Assessment of the Reliability of Fault-Tolerant Software: a Bayesian Approach. Proc. 19th International Conference on Computer Safety, Reliability and Security, SAFECOMP'2000: to appear, Rotterdam, the Netherlands 2000.

[Littlewood 2000c] Littlewood B., Popov P., Strigini L.: N-version design Versus one Good Version. Proc. International Conference on Dependable Systems & Networks (FTCS-30, DCCA-8) - Fast Abstracts: B42-B43, New York, USA, 2000.

[Littlewood 2000d] Littlewood B., Popov P., Strigini L., Shryane N.: Modelling the effects of combining diverse software fault removal techniques. IEEE Transactions on Software Engineering: to appear, 2000.

[Littlewood 2000e] Littlewood B., Strigini L. A discussion of practices for enhancing diversity in software designs, DISPO project technical report, Centre for Software Reliability, City University, 2000.

[Littlewood 2001] Littlewood B., Popov P., Strigini L.: Modelling software design diversity - a review. ACM Computing Surveys: to appear, 2001.

[Popov 1998a] Popov P., Strigini L., Pizza M. Diverse redundancy against design error: a model of fault creation and its implications on reliability, DISPO/DISCS projects technical report, Centre for Software Reliability, City University, 1998.

[Popov 1998b] Popov P.T., Strigini L.: Conceptual models for the reliability of diverse systems - new results. Proc. 28th International Symposium on Fault-Tolerant Computing (FTCS-28): 80-89, Munich, Germany 1998.

[Popov 1999a] Popov P., Strigini L. The reliability of diverse systems: a contribution using modelling of the fault creation process, DISPO/DISCS projects technical report, Centre for Software Reliability, City University, 1999.

[Popov 1999b] Popov P., Strigini L., May J., Kuball S. Estimating Bounds on the Reliability of Diverse Systems, DISPO project technical report, Centre for Software Reliability, City University, London, 1999.

[Popov 1999c] Popov P., Strigini L., Romanovsky A.: Choosing effective methods for design diversity - how to progress from intuition to science. Proc. SAFECOMP '99, 18th International Conference on Computer Safety, Reliability and Security: 272-285, Toulouse, France 1999.

[Popov 2000] Popov P., Strigini L., Littlewood B.: Choosing between Fault-Tolerance and Increased V&V for Improving Reliability. Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Monte Carlo Resort, Las Vegas, Nevada, USA 2000.