

# Complexity Is the Enemy of Dependability – Can Diversity Provide a Defence?

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

Complexity is the enemy of dependability. If we want to build systems that are safe and reliable, we know that we should make them as simple as possible. And even then our task may not be an easy one.

Unfortunately, not all complexity is the result of poor design. Sometimes complexity is necessary – the inevitable consequence of a reasonable need for extensive functionality. In such circumstances simplicity may not be achievable. How, then, do we make our systems dependable?

One way forward is through the use of diversity. The best-known applications of diversity in computing date back a couple of decades and involve design diversity, e.g. n-version programming. This kind of diversity, in the pursuit of fault tolerance, has had a checkered history. On the one hand, several famous experiments have shown that the benefits fall far short of what might be expected if the different versions were to fail independently. On the other hand, several serious industrial applications seem to have worked well.

The reliability of a fault tolerant system, say a 1-out-of-2 safety protection system, will depend upon a trade-off between the reliabilities of the individual versions, and the dependence between them. Informally, the more diverse they are, the less dependent their failures will be, and thus the more reliable the overall system.

It is only in recent years that we have had sophisticated formal probability models of diversity. These have brought deeper understanding of the nature of dependence, and given insights into how it is best to go about deploying diversity in system design.

The major thesis of this talk, though, is that diversity is ubiquitous – almost wherever you look you will find it. In particular, I shall show that diversity appears to be ‘a good thing’ not only for fault-tolerant system design but in other areas of software engineering. An example is the use of diverse methods to seek faults in software. Another is in diverse arguments – e.g. multi-legged safety cases – to gain greater confidence in dependability claims.

I shall try to show that the formalism developed for design diversity will work in these wider contexts. Some of the results are intuitively plausible, some of them are rather surprising and non-intuitive. Altogether, it seems that this is another area of computer science where mathematical formality can bring better understanding and, ultimately, better control over the attributes of the systems we build.