

# Choosing Between Fault-Tolerance and Increased V&V for Improving Reliability

Peter Popov, Lorenzo Strigini, Bev Littlewood  
Centre for Software Reliability, City University, London, UK  
phone: +44 207 477 8963, fax: +44 207 477 8585  
{ptp, strigini, b.littlewood}@csr.city.ac.uk

(Presented at PDPTA'2000, 26 - 29 June, 2000, Las Vegas, USA.)

This version corresponds to Version 1.0, 31 May 2000, of the DISPO Technical Report of the same title, PP\_DD\_TR-06\_v1.0)

## Abstract

Fault tolerant systems based on the use of software design diversity may be able to achieve high levels of reliability more cost-effectively than other approaches, such as heroic debugging. Earlier experiments have shown that multi-version software systems are more reliable than the individual versions. However, it is also clear that the reliability benefits are much worse than would be suggested by naive assumptions of failure independence between the versions.

To decide whether to use design diversity or other means for achieving the desired reliability a developer would need to know how they compare from the viewpoint of cost-effectiveness. Empirical data are insufficient for deciding this question, and expert opinions differ. We refute a recently published argument in favour of diversity and in the process show some general factors deciding whether process improvement, or debugging of the versions in a multiple-version system, will increase or decrease the statistical correlation between failures of the versions. The conclusion is that there is as yet no evidence that the choice between design diversity and other means of reliability improvement can be decided by general arguments rather than by detailed (and uncertain) special-case analysis.

**Keywords:** Software diversity, Fault-tolerance, Software Reliability Growth, Failure Dependence, multiple-version software

## 1. Introduction

The use of software design diversity in a fault tolerant architecture has been suggested and at times adopted as one solution to the problem of achieving very high reliability. Whilst the benefits of software diversity do not seem to be as high as those sometimes claimed for hardware redundancy [1], there is nevertheless evidence that fault tolerance can deliver levels of reliability higher than achieved with a single software version [2]. Indeed, there are several examples of apparently successful use of design diversity in engineered systems that are in operational use, e.g. [3], [4], [5].

The reason that software fault tolerance does not deliver dramatically high reliability is that the failures of different software versions in a fault tolerant system cannot be assumed to be independent [6], [1], [7].

Copyright © 2000 by CSREA Press

Copying and reprint permission: Copying without fee is permitted provided that the copies are not made or distributed for direct commercial advantage, and credit to the source is given. Abstracting is permitted with credit to the source. Contact the editors or the publisher, for other copying, reprint or republication permission.

One reason for this is that human designers and builders of software systems tend to make similar mistakes. This means that even two versions of a program that have been produced using ‘independent’ development teams will not fail independently - instead there will be a greater tendency for the versions to fail simultaneously, thus limiting the reliability of a 1-out-of-2 system built from them. This, however, is not a serious objection against using design diversity. Design diversity would still be a reasonable use of project resources if it delivered a substantial (though short of failure independence) increase in reliability, greater than would be delivered by spending the same amount on other ways of improving reliability.

The idea that diversity may be a more cost effective way to deliver high reliability is widely shared and recently it was spelled out by Hatton [8]. A summary of Hatton’s analysis is as follows. He quotes two items of empirical data: that, in a certain experiment, 2-out-of-3 multiple-version systems delivered a probability of failure 45 times lower than the average probability for individual versions; and that (he claims) “state of the art” development processes (costing much more than “ordinary” ones) currently deliver an improvement in probability of failure of a factor of 10 over “ordinary” processes. The problem is how to generalise from these two data points (which do not refer to the same experimental conditions) to argue for a universal decision rule. In order for the two alternative solutions to be comparable either we assume that their costs are the same and compare their reliability, or fix the reliability and compare their costs. Fixing the reliability of the solution is not possible - we can measure it only after the development. Operating under the constraints of a fixed cost is a realistic situation: this is the scenario discussed by Hatton and assumed in the rest of the paper. Once the cost is fixed two ratios of probabilities of failures affect the decision which alternative is better: i) ‘state-of-the-art’ single channel vs. ordinary channels, ii) fault-tolerant software consisting of ordinary channels vs. individual ordinary channels. With the values 10 and 45, respectively, the fault-tolerant solution is the winner: it delivers better reliability at the same cost. Clearly if the numbers are such that the first ratio is greater than the second one, the solution using the ‘state-of-the-art’ version will be more cost-effective. Hatton addresses this issue by arguing that the advantage of fault tolerance over “state of the art” processes increases with higher version reliability, i.e., is more pronounced when we try to satisfy higher reliability requirements, and will in any case increase with the general on-going improvements in software development processes. So, even if fault tolerance were not the better choice in some special cases now, these cases would become rarer with further progress in the industry. He supports this claim by arguing that the advantage of a ‘state-of-the-art’ process over “ordinary” process will remain at about 10, while the advantage of a multiple-version system over its individual versions is going to increase with increases in version reliability, as it does in the special, simple case in which the versions fail independently. The interested reader is referred to Hatton’s paper for a detailed justification of his argument.

We refute Hatton’s argument for the generalisability of the “10-vs-45” case by showing that an increase in version reliability is not necessarily going to increase the advantage of multiple-version systems over their constituent versions. The next section presents a model which (among other implications) refutes Hatton’s argument.

## **Main result**

Here we give a simplified model of the effect of reliability growth on the gain from a fault-tolerant architecture. The model embodies two distinct concepts: i) modelling the software development prior to integrating the versions in a fault-tolerant architecture, and ii) explicit modelling of the activities purposed at improving the versions’ reliability after their integration in a fault-tolerant system. One possible interpretation of the second concept, which for brevity we call ‘V&V’, is assuming that it begins when the versions have passed the acceptance criteria set for the versions to be used in a particular fault-tolerant configuration. Any activity after the initial acceptance, e.g. system testing and removing faults revealed during the testing, is modelled within the second concept.

We model the first concept using the well known model of independent development of software versions to form a fault-tolerant system, proposed by Eckhardt and Lee [6]. The second concept is

modelled by introducing a factor of improvement upon the probability of failure of a randomly selected version on each point of the demand space, which is an extension of the Eckhardt and Lee (EL) model.

EL model represents the design process as a selection from a population of software versions that could be written to the same specification. The modelled process is one in which the two development teams are kept separate, but not given different directives to 'force diversity'. Thus, creating a multi-version fault-tolerant architecture can be modelled by drawing "independently" twice (or more times if the fault-tolerant architecture employed requires more versions) from the same population (with replacement), according to a given probability distribution,  $P(\bullet)$  defined on the population of versions  $\{\pi_1, \pi_2, \dots, \pi_n\}$ .  $P(\pi)$  represents the likelihood that version  $\pi$  will be created during the development.

The failure behaviour of versions is modelled by their *scores* on each demand. The score of version  $\pi$  on demand  $x$ ,  $\omega(\pi, x)$ , takes the value 1 if  $\pi$  fails on  $x$  and 0 if it does not. A difficulty function,  $\theta(x)$ , on each point,  $x$ , of the demand space for the population of versions  $\{\pi_1, \pi_2, \dots, \pi_n\}$  with a probabilistic measure  $P(\bullet)$  can be defined as follows:

$$\theta(x) = \sum_{\pi} \omega(\pi, x) P(\pi) \quad (1)$$

The harder the demand  $x$  is for the developers to deal with properly, the greater the value of  $\theta(x) \in [0,1]$ .

Clearly, the probability of failure (on a randomly selected demand) of a *randomly selected version*  $\Pi$ , (for a given demand profile  $Q(\bullet)$ , i.e. demand  $x$  will be input to the software in operation with probability  $Q(x)$ ) will be:

$$\begin{aligned} \Pr(\Pi \text{ fails}) &= \\ &= \sum_{\pi} \sum_x \omega(\pi, x) P(\pi) Q(x) = \sum_x \theta(x) Q(x). \end{aligned} \quad (2)$$

From [6] we know that the probability of failure of a 1-out-of-2 system on a randomly selected demand is:

$$\begin{aligned} \Pr(\Pi_1, \Pi_2 \text{ fail}) &= \\ &= P(\Pi_1 \text{ fails}) P(\Pi_2 \text{ fails}) + \text{var}(\Theta), \end{aligned} \quad (3)$$

where  $\Theta$  denotes the difficulty function on a randomly selected demand. If there is no variability of the difficulty over the points of the demand space, i.e.  $\theta(x) = \text{const}$ , versions 'on average' fail independently. In this case the calculation of the gain from employing a two channel system over a single channel is straightforward. The difficulty however is unlikely to be constant. In practice the variance plays an important role; it is usually the main factor of system unreliability,

$$\text{var}(\Theta) > \Pr(\Pi_1 \text{ fails}) \Pr(\Pi_2 \text{ fails}).$$

The effect of the reliability growth on the population of versions is modelled as follows. Assume that the probability of failure of a randomly selected version on a particular demand,  $x$ , will be reduced by a factor of  $k(x)$  after the V&V are undertaken. That is, we represent the difficulty function after the growth of reliability as  $\theta_{\text{post}}(x) = \theta(x)k(x)$ .

The effectiveness of the V&V on a randomly selected demand is described thus by a random variable,  $K$ , taking the value  $k(x)$  on demand  $x$ . The values  $k(x)$  may vary between demands representing the variation of the effectiveness of the V&V over the points of the demand space.

The probability of failure of a randomly selected version subjected to V&V is:

$$\begin{aligned} \Pr(\Pi \text{ fails} | V \& V) &= \sum_x \theta_{\text{post}}(x) Q(x) = \\ &= E[\Theta K] = E[\Theta] E[K] + \text{cov}(\Theta, K) \end{aligned} \quad (4)$$

Then, assuming that the independence of version failures conditional on  $x$  is still in place after the V&V<sup>1</sup> we can express the probability of coincident failure of two randomly selected versions as:

$$\begin{aligned}
\Pr(\Pi_1, \Pi_2 \text{ fail} | V \& V) &= \sum_x [\theta_{\text{Post}}(x)]^2 Q(x) = \\
&= E[\Theta^2 K^2] = E[\Theta^2] E[K^2] + \text{cov}(\Theta^2, K^2) = \\
&= \left[ (\Pr(\Pi \text{ fails}))^2 + \text{var}(\Theta) \right] \left[ (E[K])^2 + \text{var}(K) \right] \\
&\quad + \text{cov}(\Theta^2, K^2).
\end{aligned} \tag{5}$$

Now we can express the ratio of the probabilities of failure on demand (*pdf*) of a randomly selected single version and of a 1-out-of-2 system after the V&V:

$$\begin{aligned}
\frac{\Pr(\Pi \text{ fails} | V \& V)}{\Pr(\Pi_1, \Pi_2 \text{ fail} | V \& V)} &= \\
&= \frac{E[\Theta] E[K] + \text{cov}(\Theta, K)}{\left[ (\Pr(\Pi \text{ fails}))^2 + \text{var}(\Theta) \right] \left[ (E[K])^2 + \text{var}(K) \right] + \text{cov}(\Theta^2, K^2)}
\end{aligned} \tag{6}$$

Despite its complexity, (6) allows us to analyse easily some special cases:

1. Assume that the effectiveness of the V&V is constant on all points of the demand space,  $k(x) = K_V$  = const. Then, clearly,  $E[K] = K_V$ ,  $\text{var}(K) = 0$ ,  $\text{cov}(\Theta, K) = 0$  and  $\text{cov}(\Theta^2, K^2) = 0$ , hence:

$$\begin{aligned}
\frac{\Pr(\Pi \text{ fails} | V \& V, K_V)}{\Pr(\Pi_1, \Pi_2 \text{ fail} | V \& V, K_V)} &= \frac{E[\Theta] K_V}{\left[ (\Pr(\Pi \text{ fails}))^2 + \text{var}(\Theta) \right] \left[ (K_V)^2 \right]} = \\
&= \frac{E[\Theta]}{(K_V) \left[ (\Pr(\Pi \text{ fails}))^2 + \text{var}(\Theta) \right]} = \frac{\Pr(\Pi \text{ fails})}{(K_V) \Pr(\Pi_1, \Pi_2 \text{ fail})}.
\end{aligned} \tag{7}$$

In the general case  $K_V < 1$  (otherwise there is no growth of reliability). Therefore:

$$\begin{aligned}
\frac{\Pr(\Pi \text{ fails} | V \& V, K = K_V)}{\Pr(\Pi_1, \Pi_2 \text{ fail} | V \& V, K = K_V)} &= \\
&= \frac{\Pr(\Pi \text{ fails})}{(K_V) \Pr(\Pi_1, \Pi_2 \text{ fail})} > \frac{\Pr(\Pi \text{ fails})}{\Pr(\Pi_1, \Pi_2 \text{ fail})}
\end{aligned} \tag{8}$$

In other words, if the effectiveness of V&V does not vary between demands, then the gain from the fault-tolerant architecture increases with the reliability growth. This special case does not seem very realistic, however. In the general case  $K$  will vary between the demands, [9].

2. Assume that the V&V applied is such that as a result all versions will perform correctly everywhere except on a single demand  $x^*$ . Formally,  $k(x^*)=1$ , and  $k(\forall y \neq x^*)=0$ . To make calculations easier, assume that  $\theta(x^*)=1$ , i.e. every possible version fails on  $x^*$  prior to the V&V. Clearly, after the V&V, all versions will be identical with respect to their failure behaviour - they all will fail on the same demand,  $x^*$ . Thus, after the V&V the fault-tolerant architecture will be *no better* than each of the versions from the population, hence:

$$\frac{\Pr(\Pi | V \& V)}{\Pr(\Pi_1, \Pi_2 | V \& V)} = 1.$$

This is the worst possible scenario but it is *not unrealistic*. A fault in the requirements specification may

<sup>1</sup> This assumption is known to apply in the case of independent testing of versions with the same testing profile, although it may be not true in other cases, e.g. back-to-back testing.

well produce this scenario. Initially wrong specifications, well understood by the development teams and correctly implemented, will make all versions fail on demands, related to the faulty part of the specs, hence  $\theta(x^*) = 1$ . Testers, using the requirements specification as guidelines for generating test cases, will *consistently let the fault go unnoticed*, hence  $k(x^*) = 1$ . The point here is not that this extreme scenario is common. The point is that *it is possible* and if this happens diversity buys us nothing.

The second scenario shows that the ratio of the probabilities of failure of an average single ordinary version vs. an average fault-tolerant system can be any number greater or equal to 1. If the ratio of the probabilities of failure of average ordinary and average 'state-of-the-art' software is greater than 1, which is plausible, then employing diversity will result in a worse system than using a single 'state-of-the-art' version. In other words, we have shown that the number 45 calculated by Les Hatton is only one possibility number which by chance is greater than the ratio of the probabilities of failure of ordinary and 'state-of-the-art' software, assumed to be equal to 10. According to our development, 45 could have been any number greater than 1 and had this been lower than 10 the conclusions would have been that the 'state-of-the-art' is better. The main question, therefore, is how likely it is in practice that the growth of reliability will decrease the gain from diversity as suggested by our second scenario. And the answer is "we do not know".

The two special cases show two different trends of the evolution of the gain from employing software fault-tolerance over a single version software as a function of reliability growth. In the first case, increasing the versions' reliability by using V&V increases the gain. Thus, recommendations to use diversity with better versions seem justified. The second special case, however, is a warning against recommending diversity as a most cost-effective solution for achieving high reliability. It is *possible* that the hardly earned benefits from diversity be completely lost by subjecting the system to further V&V. The versions may become identical from the viewpoint of their failure behaviour: in this case there will be no justification of developing two (or more) channels.

This model was intended to represent the growth of the versions' reliability due to V&V. We believe, however, it is applicable in a more general context - to represent two different processes, A and B, of different quality, e.g. a current process and a future one produced by improvements in practice, not necessarily the evolution of the same software versions. A decision maker can be faced with the following situation: (s)he can purchase two programs from vendors known to use an ordinary-quality process (producing products with modest reliability) or purchase a single product from a vendor known for a very high quality process (which generally delivers exceptionally reliable products). The difficulty functions,  $\theta_A(x)$  and  $\theta_B(x)$ , respectively, which represent the two development processes, will differ.

The probabilities of failure on demand of the versions created using development processes A and B will be:

$$\Pr(\Pi_A \text{ fails}) = \sum_x \theta_A(x)Q(x) \text{ and}$$

$$\Pr(\Pi_B \text{ fails}) = \sum_x \theta_B(x)Q(x), \text{ respectively.}$$

We can describe this difference as:

$$\theta_A(x) = \theta_B(x)k(x). \tag{9}$$

There is no reason in this case for us to assume that  $k(x) < 1$  on each  $x$  of the domain space. The fact that A is a better process than B only requires the following to be true:

$\Pr(\Pi_A \text{ fails}) \leq \Pr(\Pi_B \text{ fails})$ , which still allows us to apply the math developed in this section to the improved process expressing its difficulty function using (9). Therefore, even with the better process (due to using improved practices) the mathematical development in this section will hold, hence the conclusion that the ratio (6) is not guaranteed to increase with the better process will hold, too.

## Discussion

Is the second special case more than a mathematical curiosity? To the best of our knowledge, there have never been systematic experiments trying to capture the trend of the ratio given by (6), at least nothing exists in the public domain. Such experiments would be similar to the experiments carried out by Knight and Leveson in the 80's [1] and by several universities in the USA, described in [10]. The difference is that instead of single snapshot of the versions' reliabilities (taken during the system testing) several similar snapshots will be required as the testing progresses.

At least two public domain publications are related to the issue: [11] and [10]. In the first publication the authors analysed the effect of the increased reliability of the population (by taking out one by one the least reliable of the 27 versions) on the correlation between version failures. These results were used by Hatton to conclude that reliability improvement favours diversity as a more cost-effective approach for achieving high system reliability than a single 'state-of-the-art' version. The second large scale experiment resulted in creating 20 functionally identical software versions [10], which were tested in different environments (some failures of the sensors, the readings of which were used in the calculations, were simulated). In different environments the versions showed different reliability, which can be used for the purposes of this study. These data revealed that with the increase of reliability the ratio (6) decreased, which is as predicted by our special case 2. Similar results were reported by Djambazov and Popov, [12]. These fragmented data do show that both trends described above are possible. Using a more detailed model of plausible mechanisms of fault creation [13] we arrived at similar conclusion.

## Conclusion and open questions

Using a simple model we showed that the effect of the reliability growth on the dependence between the failures of diverse software versions can go in two opposite directions: the gain from employing software fault-tolerance can increase or decrease with reliability growth. Currently the factors which make either of the two possible trends more likely are *unclear*. Among other consequences, this invalidates Hatton's argument in [8].

The issue needs studying both theoretically, via modelling, and empirically, by conducting focussed experiments similar to those conducted in the USA a decade ago, in which instead of taking a single snapshot of the population by subjecting it to extensive system testing, a multitude of such snapshots are necessary combined with successive repairs of the discovered faults. Multiple snapshots of the evolution of the populations of versions and the population of fault-tolerant systems will allow to see the evolution of the ratio 'average single version vs. average fault-tolerant system'. Each experiment of this kind will provide a single data point. Repeating the experiment in various development environments will provide the necessary statistical material for more definitive conclusions.

If the experiments confirm that both trends may indeed occur, then it is essential to study the constraints, which, if imposed on the development process, will limit the possibly decreasing trend of the gain from using diverse software to values beyond the ratio 'ordinary vs. state-of-the-art software'. Only knowing these constraints will allow us to recommend as cost effective a particular process to be used in the development of diverse software. Currently such constraints are unknown and therefore the unconditional recommendation to use diversity is not justifiable.

## References

- [1] J. C. Knight and N. G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming", IEEE Transactions on Software Engineering, SE-12, pp. 96-109, 1986.
- [2] J. C. Knight and N. G. Leveson, "An empirical study of failure probabilities in multi-version software", in Proc. 16th International Symposium on Fault-Tolerant Computing (FTCS-16), Vienna, Austria, 1986, pp. 165-170.
- [3] U. Voges (Ed.), "Software diversity in computerized control systems", Wien, Springer-Verlag, 1988.

- [4] D. Briere and P. Traverse, "Airbus A320/A330/A340 Electrical Flight Controls - A Family Of Fault-Tolerant Systems", in Proc. 23rd International Symposium on Fault-Tolerant Computing (FTCS-23), Toulouse, France, 22 - 24, 1993, pp. 616-623.
- [5] H. Kantz and C. Koza, "The ELEKTRA Railway Signalling-System: Field Experience with an Actively Replicated System with Diversity", in Proc. 25th IEEE Annual International Symposium on Fault -Tolerant Computing (FTCS-25), Pasadena, California, 1995, pp. 453-458.
- [6] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors", IEEE Transactions on Software Engineering, SE-11, pp. 1511-1517, 1985.
- [7] B. Littlewood and D. R. Miller, "Conceptual Modelling of Coincident Failures in Multi-Version Software", IEEE Transactions on Software Engineering, SE-15, pp. 1596-1614, 1989.
- [8] L. Hatton, "N-Version Design Versus One Good Version", IEEE Software, 14, pp. 71-76, 1997.
- [9] B. Littlewood, P. Popov, L. Strigini and N. Shryane, "Modelling the effects of combining diverse software fault removal techniques", IEEE Transactions on Software Engineering, pp. to appear.
- [10] D. E. Eckhardt, A. K. Caglayan, J. C. Knight, L. D. Lee, D. F. McAllister, M. A. Vouk and J. P. J. Kelly, "An Experimental Evaluation of Software Redundancy as a Strategy for Improving Reliability", IEEE Transactions on Software Engineering, 17, pp. 692-702, 1991.
- [11] J. C. Knight, N. G. Leveson and L. D. S. Jean, "A Large Scale Experiment in N-Version Programming", in Proc. 15th Int. Symp. on Fault Tolerant Computing (FTCS-15), Ann Arbor, Michigan, USA, 1985, pp. 135-139.
- [12] K. B. Djambazov and P. Popov, "The effects of testing on the reliability of single version and 1-out-of-2 software", in Proc. 6th Int. Symposium on Software Reliability Engineering, ISSRE'95, Toulouse, 1995, pp. 219-228.
- [13] P. Popov, L. Strigini and M. Pizza, "The efficacy of diverse redundancy against design error: some practical considerations", in Proc. INucE Third International Conference on Control and Instrumentation in Nuclear Installations, Edinburgh, U.K., 1998.
- Also [www.csr.city.ac.uk/csr\\_city/projects/diversity/](http://www.csr.city.ac.uk/csr_city/projects/diversity/)