N-version design Versus one Good Version

Bev Littlewood, Peter Popov and Lorenzo Strigini
Centre for Software Reliability, City University
Northampton Square, London, EC1V 0HB

(Presented at DSN'2000, 25 - 28 June, 2000, New York, USA.
This version corresponds to Version 1.0, 31 May 2000, of the DISPO Technical Report of the same title)

Software Diversity as a way of achieving high reliability of software

Software diversity has long been seen as way of achieving higher reliability of software than is attainable by a single software version subjected to a heroic testing.

The known experiments with software diversity confirm that indeed fault-tolerant software employing diversity is "on average" more reliable than a single software version. Utilising design diversity when high reliability is required is, nevertheless, problematic. The cost of diverse system, which may increase substantially, is not the only obstacle. The main problem is assessing how much gain in reliability has actually been achieved. It was shown empirically and theoretically that even versions developed by not communicating development teams tend to fail simultaneously more often than they should under if their failures were independent. Quantifying the dependence between failures is no easier than assessing the reliability of the system, very difficult, because the failures of such systems are very rare and putting a reasonable confidence in the assessment results requires a very long observation.

The idea that diversity may be a more cost effective way to deliver high diversity is alive and recently it was spelled out by Hatton [Hatton 1997]. His main point, which seems to be shared by many, is that despite the dependence between the failures of the independently developed software versions, for a given target of the system reliability fault-tolerant software, e.g. 2-out-of-3 system, may be a more cost effective solution than a single software version. The key idea is that despite the dependence between the failures of the versions, a fault-tolerant system consisting of versions of 'ordinary' quality will deliver better reliability than if the efforts to develop several channels were used to produce a single 'state-of-the-art' version. Hatton concludes that taking into account the ratio between the reliability of the 'ordinary' and the 'state-of-the-art' software favours the decision to develop diverse software. The interested reader is referred to Hatton's paper for detailed justification of this idea. He goes even further to conclude that if the targeted reliability increases, than the cost benefits of using design diversity increase, too. The implications of the analysis are that when the developer can afford to produce several versions independently this is the way to go - the reliability of the fault-tolerant system is likely to be better than if a single version were developed.

We scrutinise Hatton's analysis, which crucially depends on the assumption that the ratio between the probability of failure of a single channel and a 2-out-of-3 (or any other fault-tolerant architecture) will increase even though slower that if failures were independent. We show, using data reported by others, that this assumption may be wrong.

Empirical data available to date

The first source of evidence to support the idea that software diversity may be a cost effective way to deliver high reliability of software is the data from the well known experiment by Knight and Leveson [Knight *et al.* 1985]. In this experiment 27 versions were developed to the same specification and then subjected to 1 000 000 tests. Their probabilities of failure ware estimated, which is summarised in Table 1 and 2.

Removing versions with worse reliability emulates improving the quality of the development process. Removing the best versions, on the other hand, emulates the worsening of the development process. By inspecting the effect of the 'improving/worsening' of the process on the ratio of the probabilities of failure of a single and a 2-out-of-3 system we can confirm the thesis developed by Hatton: the better process leads to increase of the ratio.

Table 1

Removed	Single	2-out-of-3	Ratio
0	0.00069833	0.00003667	19
1	0.00035381	0.00003169	11
2	0.00027608	0.00002423	11
3	0.00023058	0.00002393	10
4	0.00019065	0.00002575	7
5	0.00015677	0.00001155	14
6	0.00012219	0.00000635	19
7	0.00010060	0.00000539	19
8	0.00008342	0.00000470	18
9	0.00007011	0.00000149	47
10	0.00005841	0.00000100	59
11	0.00004556	0.00000111	41
12	0.00003127	0.00000127	25
13	0.00002657	0.00000074	36
14	0.00002154	0.00000046	47
15	0.00001667	0.00000018	92
16	0.00001173	0.00000022	54
17	0.00000670	0.00000000	-
18	0.00000156	0.00000000	-
19	0.00000075	0.00000000	-

Table 2

Removed	Single	2-out-of-3	Ratio
6	0.000902	0.000060	15.0
7	0.000948	0.000066	14.4
8	0.000997	0.000073	13.6
9	0.001050	0.000081	12.9
10	0.001108	0.000090	12.3
11	0.001173	0.000097	12.0
12	0.001246	0.000104	12.0
13	0.001328	0.000119	11.1
14	0.001423	0.000127	11.2
15	0.001532	0.000148	10.4
16	0.001648	0.000173	9.5
17	0.001786	0.000208	8.6
18	0.001955	0.000243	8.0
19	0.002159	0.000240	9.0
20	0.002406	0.000299	8.0
21	0.002715	0.000267	10.1
22	0.003081	0.000199	15.6

Tables 1 and 2. *The effect of the growth/decay of population reliability on the effectiveness of software fault-tolerance: Knight and Leveson data [Knight et al. 1985].*

The second source we used are the data reported by Eckhardt et al in [Eckhardt *et al.* 1991]. In this experiment 20 versions were developed and then subjected to substantial testing: each version was tested on ~900 000 inputs. In this experiment, however, the software was placed in varying environment - some failures of sensors were simulated which affected versions reliability. We do not give here detailed account of the conditions under which the tests were conducted. The interested reader is referred to [Eckhardt *et al.* 1991] for details. We use the testing results obtained in the created 6 different states of the environment and calculate the ratio between the reliability of a single channel and of a 2-out-of-3 system (In both cases as in the previous example we use the averages, calculated on the populations of all single versions and all possible 3 version systems). The results are summarised in Table 3.

Table 3

	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$
single channel pfd, P_1	0.000073	0.000472	0.000038	0.006387	0.000083	0.028928
$P_{2\text{-out-of-3}}$	1.70E-05	1.40E-04	1.80E-05	1.00E-04	1.70E-05	2.40E-03
P_1/ $P_{2\text{-out-of-3}}$	4.3	3.38	2.13	63.87	4. 89	12.05

It is obvious that this case reveals a patters which differs from the pattern we observed in Table 1 and 2: despite the fluctuations, when the reliability increases, the ratio decreases, i.e. the gain from the 2-out-of-3 is lower when the versions are more reliable.

Clearly, the experiments mentioned above were not about the effect of the reliability on the dependence between versions' failures. The results presented, therefore, could be criticised for not being based on unsuitable data. We accept this criticism. The difference between the results, nevertheless, remains and is dramatic: the trends have *different signs*. Is this difference no more that a intriguing coincidence or is there something more? In a simulation study Djambazov et al. [Djambazov & Popov 1995] reported that with reliability growth the dependence between the failures of the versions increases, which is in line with the second data set.

Discussion

Our preliminary modelling of the effect of the reliability growth on the dependence between the failures of the channels showed that the dependence *can* change in both directions: it may decrease, as Hatton suggested, but it can also increase, as suggested by Table 3. It is very difficult, if possible at all, to tell in advance which of the two possible trends will be in place in a particular case. Therefore, it appears that Hatton's suggestion that design diversity is always going to be more cost effective than developing a single version software is not trustworthy. In order for us to be certain that diversity will bring more than it takes we need to measure the dependence, which is currently an open question. We need even more - to predict how the dependence will evolve with the reliability growth, which is even more difficult.

References

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

[Eckhardt *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 Transactions on Software Engineering*, 17 (7), pp.692-702, 1991.

[Hatton 1997] L. Hatton, “N-Version Design Versus One Good Version”, *IEEE Software*, 14 (6) 1997.

[Knight *et al.* 1985] J. C. Knight, N. G. Leveson and L. D. S. Jean, “A Large Scale Experiment in N-Version Programming”, in *15th Int. Symp. on Fault Tolerant Computing (FTCS-15),* Ann Arbor, Michigan, USA, pp.135-139, IEEE Computer Society Press, 1985.