

# Multi-Version Software versus One Good Version: A Further Study and Some Results

Paul Townend      Jie Xu      Malcolm Munro  
Dept. of Computer Science, University of Durham, DH1 3LE, England  
{P.M.Townend, Jie.Xu, Malcolm.Munro}@durham.ac.uk

## Abstract

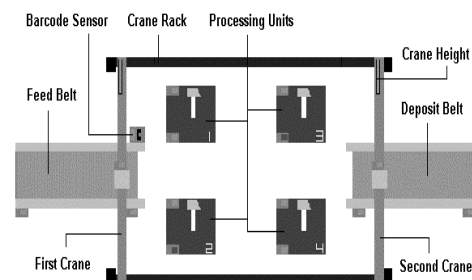
Based on our previous results, we describe a further study and discuss some new findings in order to establish whether or not the multi-version method can offer increased dependability over the traditional single-version development approach when given the same level of resources.

## 1. Introduction

An increasing range of industries has a growing dependence on software-based systems, many of which are safety-critical, real-time applications that require extremely high dependability. Multi-version programming has been proposed as a method for increasing the overall dependability of such systems – however, the increased cost of using this approach may mean that this increase in dependability is not worth the extra expense involved. In a recent study [TOW01], the authors investigated for the first time whether or not the multi-version approach can result in a more dependable system than “traditional” single-version approaches, when given a fixed budget.

This was performed by implementing both a multi-version system and a single-version system to control the simulation of a factory production cell [LOT96]. The factory production cell simulation was chosen as it was necessary to implement an application that was both small enough to construct within the resource budget allocated, and also complex enough to have the potential for faults to be present within the system code. It was also desirable for the application to be a real-time system, as such systems invariably involve high reliability and safety requirements, as well as high timing constraints. The production-cell, detailed in Figure 1, consists of two conveyor belts, one of which delivers the raw units (blanks) into the system, and one of which moves the blanks out of the system once they have been fully processed. The unit also consists of four separate processing workstations. Two cranes are used to transport blanks around the system. Each blank has its own bar-code, which will identify which workstations it needs to be placed in, and the minimum and maximum amounts of time that it could spend within each workstation. Blanks may be processed either in specific

(preserved) order (i.e. P order), or in any (non-preserved) order (i.e. NP order), depending on the instructions in the bar-code.



**Figure 1** The flexible production cell [LOT96]

The development budget was measured purely in terms of time to develop, and was split into three equal time blocks. The first time block was used by each programmer to develop a working version of the controller software; at the end of this phase, three programs were used as channels in the multi-version system. The second and third time blocks were then used by each programmer to refine their respective versions and improve their dependability as much as possible; these improved systems were then treated as single-version systems.

These additional time blocks resulted in each of the single-version systems having had the same amount of resources spent on them as the 3-version voter system. At the conclusion of the third time block, an extensive testing plan was implemented in order to ascertain the dependability of the single-version systems and the multi-version system, and an analysis was then performed to determine which of the methods had resulted in the most dependable software.

The results of the study revealed that although the single-version system was much more reliable than any individual channel of the multi-version system, the multi-version system could be considered the *safer* of the two approaches as fewer *undetected* incorrect outputs were produced. Although these results could not be considered conclusive in the general sense, they suggested that regarding the single-version method as a “seem-to-be” safer design decision for safety-critical applications was generally not justifiable.

No. of Blanks	P x P Failures		NP x NP Failures		P x NP Failures		NP x P Failures		Totals	
	No.	%	No.	%	No.	%	No.	%	No.	%
1	0/64	0	0/64	0	-	-	-	-	0/128	0
2	20/144	13.89	17/144	11.81	13/144	9.03	17/144	11.81	70/576	12.15
									70/704	9.94

**Figure 2** Results of testing the new single version system with multiple blanks (inputs)

This experiment was the first of its kind, and as such had no similar work to base itself on. Although the results are extremely interesting, the experiment had a number of weaknesses. The most major of these was that the single-version systems were each based upon a multi-version channel. Therefore, the extra development time afforded to each single-version system was used primarily to refine existing code, whereas in the real world, a larger development time may have allowed for more time to be spent on the system design, formal specification, etc. This could in turn have affected the dependability of the final single-version systems.

## 2. The Follow-up Study

In order to rectify this problem, the authors have performed a follow-up study involving the development of a single-version controller system independent of any multi-version channels, and implemented by an independent programmer. The new system was allocated an identical period of development time, and on completion of this, a new series of tests were performed on the multi-version and single-version systems from the prior experiment, and the new single-version system. In order to test the systems more accurately, tighter timing constraints were imposed upon all tests in order to highlight any weaknesses in each system.

## 3. Results

Figure 2 above details the results of the series of tests performed on the new single-version system. As can be seen, the single-version system had an overall failure rate of 9.94%. Although this is unrealistically high for a critical system, it compares favourably with the 55.96% chance of failure that the new tests caused the multi-version system, and at first glance it would appear that the dependability, the reliability in particular, of the single-version system is much greater than that of the multi-version system.

However, when the *safety* attribute of dependability is investigated, the picture changes. Safety can be measured as the likelihood of a failure remaining undetected by the control system, such as a metal 'blank' being processed incorrectly and output from the system as normal, or a blank being left in the system.

When each system failure was analyzed further, it was found that the new single-version system had an undetected failure rate of 3.27%. When the multi-version system was analyzed with the new test results, *no*

*undetected failure was discovered*. This implies that although the *reliability* of the single-version system is greater than that of the multi-version system, the multi-version system is the *safer* of the two. This agrees completely with the authors' previous findings.

## 4. Conclusions

The purpose of experiment has been to follow-up the initial study in [TOW01] and address some of its weaknesses. The major finding that has resulted is that even when the single-version system is implemented 'from scratch', independently of the multi-version channels, it can still be regarded as potentially less safe than the multi-version system, although as with the previous study, the single-version system was much more *reliable* than the multi-version system.

This is despite the tests performed in this latest study being more 'rigorous' than those of the prior study, as timing constraints were taken into consideration during testing, and failures were allowed to occur for a system failing to process a blank within a required amount of time. This was not taken into account previously.

Therefore this follow-up study has confirmed the results of [TOW01] and it can be concluded that *for this experiment*, the single-version approach still *cannot be regarded as a "seem-to-be" safer solution for safety-critical systems*.

The authors intend to carry out further work on this problem; to begin with, automated testing and fault injection techniques will be used to stress test all of the existing systems, whilst new systems are to be constructed using an improved requirements document.

## Acknowledgments

The authors would like to thank John Cullen, Alastair Davies and Ketan Mistry for their contributions to the experiments.

## References

- [LOT96] A. Lötzbeyer and R. Mühlfeld, "Task Description of a Flexible Production Cell with Real Time Properties," ([ftp://ftp.fzi.de/pub/PROST/projects/korsys/task\\_descr\\_flex\\_2\\_2.ps.gz](ftp://ftp.fzi.de/pub/PROST/projects/korsys/task_descr_flex_2_2.ps.gz)), 1996.
- [TOW01] P. Townend, J. Xu and M. Munro, "Building Dependable Software for Critical Applications: Multi-version Software versus One Good Version," to appear in *Proc. 6<sup>th</sup> IEEE International Workshop on Object-oriented Real-time Dependable Systems*, IEEE CS Press, Rome, 2001.