An Empirical Study on Reliability Modeling for Diverse Software Systems

Xia Cai and Michael R. Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong
{xcai, lyu}@cse.cuhk.edu.hk

Abstract

Reliability and fault correlation are two main concerns for design diversity, yet empirical data are limited in investigating these two. In previous work, we conducted a software project with real-world application for investigation on software testing and fault tolerance for design diversity. Mutants were generated by injecting one single real fault recorded in the software development phase to the final versions. In this paper, we perform more analysis and experiments on these mutants to evaluate and investigate the reliability features in diverse software systems. We apply our project data on two different reliability models and estimate the reliability bounds for evaluation purpose. We also parameterize fault correlations to predict the reliability of various combinations of versions, and compare three different fault-tolerant software architectures.

1 Introduction

Design diversity is one of the main techniques for software fault tolerance. This approach was proposed to achieve quality and reliability of software systems by detecting and tolerating software faults during operation. Its basic idea is to employ different development teams in building different program versions independently according to one single specification [14]. During program executions, the final consensus output is either voted by multiple versions, or verified by an acceptance test, which can be one of the program versions. The multi-version programs are expected to fail with low probability of coincident failures. Although many research efforts have been conducted for investigation, experimentation, modeling and evaluation of software design diversity, it still remains a debatable approach compared with other software engineering techniques. One main reason is the lack of real world project data on collecting the features of design diversity; and the other is the failures in diverse versions may not occur independently, making it difficult to establish justifiable predictive reliability models.

Nevertheless, to attempt the modeling of reliability and fault correlations achieved in design diversity, some methods have been proposed. Eckhardt and Lee [9] proposed the first model of fault correlation for diverse systems. Later Littlewood and Miller [13] showed a conceptual model in which the reliability of a pair of versions may even be better than what is under the assumption of independence. Dugan and Lyu [7] proposed a dependability model for N-version programming to parameterize the possibility of fault correlations. Recently, Popov Strigini et al [17] further pointed out that the bounds on the reliability of multiple-version systems can be estimated by dividing the demand space of the test cases into disjoint subdomains.

In our previous work, we have conducted a software project with real-world application and engaged multiple programming teams to independently develop program versions based on an industry-scale avionics application. We investigated the features and relationship between faults uncovered in the program versions. Mutants with real faults injected have been examined to investigate software testing and fault tolerance together [16]. In this paper, we extend our previous work to coincident faults in diverse programs. We apply different reliability models on our generated mutants and evaluate their effectiveness.

This paper is organized as follows. In Section 2, we recall the different reliability models about coincident failures in diverse software systems, as well as the descriptions and experimental procedures of our project. Section 3 shows the estimation results on reliability bounds using Popov, Strigini et al's method [17]. Section 4 performs dependability modeling [7] and provides the parameters of fault correlations for various combination of mutants. Finally, a conclusion is given in Section 5.

2 Background

2.1 Reliability Models for Diverse Systems

With multiple "independently developed" program versions, one would expect that failures in a subset of the versions may be masked or at least detected; coincident failures of all versions will be less frequent than failures of any single version; and thus a multiple-version system will fail less often than a single version. One might hope that the different versions fail "independently", but in some empirical studies failures of multiple versions were positively correlated [3, 11]. Eckhardt and Lee (EL model) [9] proposed a probability model that attempts to capture the nature of failure dependency in N-version programming. EL model is based on the notion of "variation of difficulty" over the demand space. Different parts of the demand space present different degrees of difficulty, which makes the program versions built independently more likely to fail with the same "difficult" parts of the target problem. Therefore, failure independency between program versions may not be the necessary result of "independent" development when failure probability is averaged over all demands. For most situations, a positive correlation between version failures may inherit from a randomly-chosen pair of program versions.

Littlewood and Miller (LM model) [13] showed, on the other hand, that the variation of difficulty could be turned from a disadvantage into a benefit with forced design diversity. *Forced* diversity may insist that different teams use different development methods, different testing schemes, different tools and languages [15]. With forced diversity, the problem which is more difficult for one team may be easier for another team, and vice versa. The possibility of negative correlation between two versions means that the reliability of a 1-out-of-2 system could be greater than it would be under the assumption of independence. Both EL and LM models are *conceptual* models because they do not support predictions for specific systems and highly depend on the notion of *difficulty* defined over the possible demand space [17].

In addition to EL and LM models, some *structural* models for design diversity were also proposed, such as [1, 4, 5, 6, 10, 20] to predict the reliability of diverse systems. For example, Tomek and Trivedi [21] proposed employing a stochastic reward net for design diversity analysis. On the other hand, [7] proposed a Markov reward model, where the Markov states and transitions represent the evolution of the system. Fault tree models are used to capture the effects of software faults and transient hardware faults together for each operation configuration. The fault tree model can be parameterized using experimental data over test cases and over test time frames, and probabilities of unrelated faults and related faults between two versions or common to all versions can be calculated from real data.

[17] studied how the conceptual model of failure generation can be applied to a specific set of versions. This model estimates the probability of failure on demand given the knowledge on subdomains in a 1-out-of-2 diverse system. Various alternative estimates for probability of coincident failures in the whole demand space as well as in its subdomains are investigated. Upper bounds and "likely" lower bounds for unreliability (probability of failure per demand) are obtained by using data from individual diverse versions. The proposed

methods are demonstrated on examples, showing how bounds estimated via subdomains may be tighter than those estimated on the whole demand space.

These models try to describe the features of fault correlation between different combination of versions, and predict the reliability of diverse systems. Empirical data are highly demanded for evaluation and cross-validation the usefulness and/or effectiveness of such reliability models.

2.2 Project Descriptions and Experimental Procedure

Motivated by the lack of empirical data, we conducted a real-world project for design diversity in the year 2002. The Redundant Strapped-Down Inertial Measurement Unit (RSDIMU) project involved more than one hundred students and 34 program versions were developed for a period of 12 weeks according to the same specification. The details of the project and development procedures are discussed in [16]. 21 out of the 34 versions were selected to create mutants, each of which was injected with a single fault identified in the testing phase. Following a systematic rule for the mutant creation process, 426 mutants were generated for testing and evaluation.

To investigate the nature and features of software failures, 1200 test cases were executed on these program versions as well as the generated mutants for evaluation test. Based on these results, a number of analysis and evaluations were conducted, including fault classification and distribution, effectiveness of code coverage and mutant coverage, and the similarities between different mutants.

In this paper, we will further engage these testing results and mutants to verify the effectiveness and accuracy of different reliability models for diverse software systems.

3 Evaluation on Popov, Strigini et al's Reliability Bounds Model

Popov, Strigini et al's model (PS model) [17] gave the upper and "likely" lower bounds for probability of failures on demand for a 1-out-of-2 diverse system. To get these bounds, complete knowledge on the whole demand space should be provided. As it is hard to obtain such knowledge, the demand space can be partitioned into some disjoint subsets, which are called *subdomains*. Given the knowledge on subdomains, failure probabilities of the whole system can be estimated as a function of the subdomain to which a demand belongs. The main idea is as follows.

For each subdomain S_i $(i = 1, \cdots, n)$, we assume that the following probabilities are known: The probability $P(S_i)$ of a random demand during software operation being drawn from S_i and the probabilities of failure (*pfds*) of A and B $(P_{A,B|S_i})$ for demands from S_i, $P_{A|S_i}$ and $P_{B|S_i}$. Then

$$P_{A,B|S_i} = P_{A|S_i} P_{B|S_i} + cov_i(\Omega_A, \Omega_B). \qquad (1)$$

The upper bound on the probability of system failure is determined as a weighted sum of upper bounds within subdomains:

$$P_{(A,B)} \leq \sum_i \min(P_{A|S_i}, P_{B|S_i}) P(S_i). \qquad (2)$$

The "likely" lower bound can be drawn from the assumption of conditional independence:

$$P_{A,B_{sub-ind}} = \sum_i P_{A|S_i} P_{B|S_i} P(S_i), \qquad (3)$$

where $P_{A,B_{sub-ind}}$ is the actual probability of coincident failures in each subdomain if the versions fail independently.

Alternative expressions for $P_{A,B}$ as the *pfd* of a 1-out-of-2 version system are given in Figure 1.

This model can be applied to real-world data collected for diverse software. The upper bound and the lower bound can be estimated for applications using Point Estimate method or Confidence Bounds method. In our experiment, we adopt Point Estimate method to illustrate the modeling results.

3.1 Prediction Results Using Our Data Set

In our experiment, we created 426 mutants from 21 program versions, where each mutant was injected with one real fault into the final program

Table 1. Alternative expressions for the pfd of a 1-out-of-2 system (from [17])

$\sum_{x \in D} \omega_A(x) \cdot \omega_B(x) \cdot P(x)$				
$P_A \cdot P_B$ (would be *pfd* in case of independence)	+	$cov(\Omega_A, \Omega_B)$ (accounts for variation of score between individual demands)		
$P_A \cdot P_B$	+	$cov(P_{A\|S_i}, P_{B\|S_i})$ (term for variation of *pfd* between subdomains)	+	$E(cov_i(\Omega_A, \Omega_B))$ (term for variation of score within each subdomain)
$\sum_i P_{A\|S_i} P_{B\|S_i} P(S_i)$ (*pfd* in case of independence in each subdomain)			+	$E(cov_i(\Omega_A, \Omega_B))$
$P_{A,B_{sub-ind}}$			+	$E(cov_i(\Omega_A, \Omega_B))$

versions passing the qualification test. Note the meaning of a mutant is different from that of a version, in the sense that a mutant is not a real final version but with faults injected manually. Here we treat each mutant, which contains only one real programming fault as a real version. From the analysis of severity of different faults, we notice that some faults can be more severe or even critical for the whole program, while others may have little influence on the program functionality. In this experiment, we only engage those mutants which passed the first 800 test cases [1] (as a qualification test set) to study the failure correlation of the diverse versions.

The RSDIMU application receives input values from redundant sensors and produces a consensus inertial measurement for avionic vehicles. The input domain for RSDIMU can be represented by various sensor failure conditions. In order to get the disjoint subdomains on the demand space, we follow the method described in [8] by dividing the 1200 test cases into 7 categories, i.e., $S_{0,0}$, $S_{0,1}$, $S_{1,0}$, $S_{1,1}$, $S_{2,0}$, $S_{2,1}$ and "others." These categories (or so-called "states") denote different situations that the number of faulty sensors prior to or during the measurement operations. For example, $S_{1,0}$, indicates the "state" of the environment with a single faulty sensor prior to testing and no more sensor failures during the testing. We add the 7th state, i.e., "others" to denote the situations other than the above 6 operational states. It represents those test cases in which the whole RSDIMU system would fail under some extreme circumstances. Although it is indicated in [8] that such situation has little chance of occurring in mission-critical diverse systems, we still consider it as a subdomain of the total test cases due to the following reasons: 1) these seven disjoint subdomains compose the whole demand space which cannot be fully represented with only six states; 2) for reliable systems, the diverse versions need to react correctly to extreme situations.

[1]Out of the 1200 test cases conducted during qualification test, the first 800 test cases were designed to test various functionality of the application, while the last 400 test cases were randomly generated according to real operational scenarios.

As stated, we use the first 800 test cases as the qualification test. All the mutants which passed the qualification test are adopted in this experiment, and each mutant is treated as a single version. We apply the remaining 400 test cases on these selected mutants. The number of failures of these mutants (belonging to different versions) with respect to the states of test case are listed in Table 2. Note that the six mutants are from different initial versions with injection of different design and programming faults.

To apply PS model, we define the hypothetical demand profiles for calculation and illustrate the effect of the demand profile on the upper bounds and lower bounds. The adjusted demand profile is shown in Table 3. The former three in Table 3 are hypothetical demand files described in [17], while the last one (DP4) is the real probability distribution in our 400 test cases. Furthermore, in order to simulate the model more accurately and realistically, we select mutants belonging to different program versions, e.g., pair (117,305), (215,382) and (382,403). We adopt Demand Profile 4 in our

Table 2. Failure data of mutants passing qualification test

Mutant ID	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$	S_{others}
117	0	0	0	0	0	0	3
215	0	0	0	1	0	0	0
223	0	0	0	0	0	0	3
305	2	1	2	0	0	0	0
382	0	0	0	8	0	0	1
403	0	0	0	0	0	0	3

analysis, which is the real probability distribution in our experiment.

Table 3. Demand Profile

	DP1	DP2	DP3	DP4
$p(s_{0,0})$	0.99	0.4	0.15	0.4
$p(s_{0,1})$	0.005	0.2	0.15	0.1175
$p(s_{1,0})$	0.003	0.2	0.15	0.14
$p(s_{1,1})$	0.001	0.1	0.15	0.085
$p(s_{2,0})$	0.0005	0.05	0.15	0.0825
$p(s_{2,1})$	0.0003	0.03	0.15	0.0275
p_{others}	0.0002	0.02	0.10	0.1475

In [17], Popov, Strigini et al discuss the use of both observed frequencies and of conservative confidence bounds as estimates of the conditional *pfds*, and favour the second alternative. Particularly in our case, according to Table 2, no failure was observed in some subdomains. Thus we adopt confidence bounds method to estimate the joint *pfds* in our experiment. Table 4 shows the 90% confidence upper bounds on *pfds* of mutants in subdomains, and Table 6 displays the lower bounds. Our testing results for upper bounds and lower bounds on joint *pfds* under four demand profiles are listed in Table 5 and Table 7, respectively.

3.2 Comparison and Discussion

The target objects engaged in our experiment and NASA 4-university experiment studied in [17] are different. In the latter, diverse versions are employed to explore the granularity of failure correlations between different pairs of versions. But in our experiment, we treat mutants as the target diverse versions, and we know the exact fault each mutant contains. This is more helpful in finding realistic features of faults and their coincidence in diverse systems. Furthermore, to make the comparison more reasonable, we only test the mutants passing the qualification test and then capture their behavior in the subsequent operation testing. For better realism, i.e., similarity with real-world multiple-version systems, we select mutants derived from different program versions.

The behavior of three pairs of mutants show three different features of fault coincidence of design diversity. For pair (117, 305), the two mutants fail differently on the seven subdomains. In this case, $P_{117,305upper}$ is tighter (smaller) than $min(P_{117}, P_{305})$ consistently for all demand profiles, although the difference between the two are insignificant under DP1. The reason behind is that the subdomains where mutant 117 performs better are those where mutant 305 performs worse, and vice versa, consistently. As the behavior of the two mutants are different in all subdomains, the covariance shown in Table 7 is a small positive number under DP1, while negative in the other three demand profiles. Thus the "likely" lower bound $P_{117,305_{sub\_ind10\%}}$ is greater than $P_{117} * P_{305}$ under DP1, but smaller under DP2, DP3 and DP4.

For the second pair of mutants (215,382), the covariance is positive under all demand profiles, indicating that the two mutants have related faults and may fail at the same subdomains. The upper bound $P_{215,382upper}$ equals to $min(P_{215}, P_{382})$ under all demand profiles, since mutant 382 performs worse than mutant 215 in all subdomains. As the correlation between the two mutants, the lower bounds with 90% confidence are always tighter (greater) than $P_{215} * P_{382}$ under all subdomains. This positive covariance case supports the concept of "variation of difficulty" between and within different demand subdomains.

Table 4. 90 percent confidence upper bounds on mutants' pfds in subdomains

Mutant ID	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$	S_{others}
117	0.0142	0.0468	0.0396	0.0637	0.0655	0.1746	0.1080
215	0.0142	0.0468	0.0396	0.1066	0.0655	0.1746	0.0376
223	0.0142	0.0468	0.0396	0.0637	0.0655	0.1746	0.1080
305	0.0327	0.0786	0.0907	0.0637	0.0655	0.1746	0.0376
382	0.0142	0.0468	0.0396	0.3446	0.0655	0.1746	0.0633
403	0.0142	0.0468	0.0396	0.0637	0.0655	0.1746	0.1080

Table 5. Upper bounds on the joint pfds under Demand Profiles

Pair		$P_{117\_90\%}$	$P_{305\_90\%}$	$\min(P_{117\_90\%}, P_{305\_90\%})$	$P_{117,305_{upper90\%}}$
(117, 305)	DP1	0.0146	0.0332	0.0146	0.0146
	DP2	0.0400	0.0626	0.0400	0.0386
	DP3	0.0715	0.0796	0.0715	0.0644
	DP4	0.0483	0.0562	0.0483	0.0379
(215, 382)		$P_{215\_90\%}$	$P_{382\_90\%}$	$\min(P_{215\_90\%}, P_{382\_90\%})$	$P_{215,382_{upper90\%}}$
	DP1	0.0146	0.0149	0.0146	0.0146
	DP2	0.0429	0.0672	0.0429	0.0429
	DP3	0.0709	0.1091	0.0709	0.0709
	DP4	0.0415	0.0656	0.0415	0.0415
(382, 403)		$P_{382\_90\%}$	$P_{403\_90\%}$	$\min(P_{382\_90\%}, P_{403\_90\%})$	$P_{382,403_{upper90\%}}$
	DP1	0.0149	0.0146	0.0146	0.0146
	DP2	0.0672	0.0400	0.0400	0.0391
	DP3	0.1091	0.0715	0.0715	0.0670
	DP4	0.0656	0.0483	0.0483	0.0417

The third pair (382,403) shows the possibility of negative covariance on DP3 and DP4. The covariance is a small negative number, and thus the lower bound is smaller than the probability under independence scenario. It indicates that with design diversity, the covariance of different versions may become a benefit instead of a disadvantage. Nevertheless, as in [17], our data also show that this situation is less likely to happen under DP1. The reason behind may be that the two mutants have correlations on some subdomains and no correlation on other subdomains, i.e., they have coincident failures on S_{others}, but no coincident failures on $S_{1,1}$. In DP1, the probability of the "independence" subdomain $S_{1,1}$ is a small number; while in other three demand profiles, the probability of $S_{1,1}$ is large enough to affect the overall correlation and make the reliability even higher than that of assuming "independence".

In order to assess whether the approach proposed in [17] is useful in practice, we need to answer the following questions:

1. "Does this method always produce tighter bounds than $P_A * P_B$ and $min(P_A, P_B)$?" From the analysis and discussion above, we can see that the confidence bounds are tighter under most circumstances except two situations: 1) one mutant performs worse than the other in all subdomains; and 2) with negative covariance, the lower bound is smaller than the probability under independent scenario.

2. "Does this method give tight enough predictions when used in practice?" To this question, we cannot give answers on the basis of our data, since in our experiment probabilities of common failure are measured directly from the number of common failures observed. The original method in [17] is meant for cases in which one can obtain estimates of failure probabilities (per subdomain) for the two versions separately, but does not have a chance of observing the two versions on the same test cases before making a prediction. Further experimental data are needed to be explored to answer this question.

Overall, the approach proposed in [17] of analyzing the behaviors of the versions by subdo-

Table 6. 90 percent confidence lower bounds on mutants' pfds in subdomains

Mutant ID	$S_{0,0}$	$S_{0,1}$	$S_{1,0}$	$S_{1,1}$	$S_{2,0}$	$S_{2,1}$	S_{others}
117	0.00065	0.00219	0.00185	0.00301	0.00309	0.00874	0.02939
215	0.00065	0.00219	0.00185	0.01529	0.00309	0.00874	0.00175
223	0.00065	0.00219	0.00185	0.00301	0.00309	0.00874	0.02939
305	0.00686	0.01113	0.01949	0.00301	0.00309	0.00874	0.00175
382	0.00065	0.00219	0.00185	0.16154	0.00309	0.00874	0.00890
403	0.00065	0.00219	0.00185	0.00301	0.00309	0.00874	0.02939

Table 7. Lower bounds on the joint pfds under Demand Profiles

Pair		$P_{117\_10\%}$	$P_{305\_10\%}$	$cov(S_{117\_10\%}, S_{305\_10\%})$	$P_{117\_10\%}P_{305\_10\%}$	$P_{117,305_{sub\_ind10\%}}$
(117, 305)	DP1	$6.73 \cdot 10^{-4}$	$6.91 \cdot 10^{-3}$	$3.86 \cdot 10^{-8}$	$4.65 \cdot 10^{-6}$	$4.69 \cdot 10^{-6}$
	DP2	$2.37 \cdot 10^{-3}$	$9.62 \cdot 10^{-3}$	$-4.26 \cdot 10^{-6}$	$2.28 \cdot 10^{-5}$	$1.86 \cdot 10^{-5}$
	DP3	$5.87 \cdot 10^{-3}$	$8.02 \cdot 10^{-3}$	$-1.80 \cdot 10^{-5}$	$4.71 \cdot 10^{-5}$	$2.91 \cdot 10^{-5}$
	DP4	$5.87 \cdot 10^{-3}$	$7.79 \cdot 10^{-3}$	$-2.47 \cdot 10^{-5}$	$4.57 \cdot 10^{-5}$	$2.09 \cdot 10^{-5}$
		$P_{215\_10\%}$	$P_{382\_10\%}$	$cov(S_{215\_10\%}, S_{382\_10\%})$	$P_{215\_10\%}P_{382\_10\%}$	$P_{215,382_{sub\_ind10\%}}$
(215, 382)	DP1	$6.80 \cdot 10^{-4}$	$8.27 \cdot 10^{-4}$	$2.39 \cdot 10^{-6}$	$5.26 \cdot 10^{-7}$	$2.95 \cdot 10^{-6}$
	DP2	$3.05 \cdot 10^{-3}$	$1.78 \cdot 10^{-2}$	$1.98 \cdot 10^{-4}$	$5.43 \cdot 10^{-5}$	$2.52 \cdot 10^{-4}$
	DP3	$4.95 \cdot 10^{-3}$	$2.76 \cdot 10^{-2}$	$2.50 \cdot 10^{-4}$	$1.37 \cdot 10^{-4}$	$3.86 \cdot 10^{-4}$
	DP4	$2.83 \cdot 10^{-3}$	$1.63 \cdot 10^{-2}$	$1.70 \cdot 10^{-4}$	$4.62 \cdot 10^{-5}$	$2.16 \cdot 10^{-4}$
		$P_{382\_10\%}$	$P_{403\_10\%}$	$cov(S_{382\_10\%}, S_{403\_10\%})$	$P_{215\_10\%}P_{382\_10\%}$	$P_{382,403_{sub\_ind10\%}}$
(382, 403)	DP1	$8.27 \cdot 10^{-4}$	$6.73 \cdot 10^{-4}$	$4.62 \cdot 10^{-7}$	$5.57 \cdot 10^{-7}$	$1.02 \cdot 10^{-6}$
	DP2	$1.78 \cdot 10^{-2}$	$2.37 \cdot 10^{-3}$	$1.61 \cdot 10^{-5}$	$4.23 \cdot 10^{-5}$	$5.84 \cdot 10^{-5}$
	DP3	$2.76 \cdot 10^{-2}$	$5.87 \cdot 10^{-3}$	$-4.86 \cdot 10^{-5}$	$1.62 \cdot 10^{-4}$	$1.13 \cdot 10^{-4}$
	DP4	$1.63 \cdot 10^{-2}$	$5.86 \cdot 10^{-3}$	$-1.16 \cdot 10^{-5}$	$9.56 \cdot 10^{-5}$	$8.40 \cdot 10^{-5}$

mains appears to help, with our project data, in revealing the features of failure correlation among diverse programs.

4 Evaluation on Dugan and Lyu's System Reliability Model

Dugan and Lyu (DL model) proposed a dependability modeling methodology for fault-tolerant software and systems [7]. The DL reliability model is constructed by three parts: a Markov model details the system structure, and two fault trees represent the causes of unacceptable results in the initial configuration and in the reconfigured degraded state. Based on this 3-level reliability model, three parameters can be estimated according to the experimental data: P_V, the probability of an unrelated fault in a version; P_{RV}, the probability of a related fault between two versions; and P_{RALL}, the probability of a related fault in all versions. The fault tree models for 2, 3 and 4 version systems are shown in Figure 1. The three parameters are calculated by the following equations for 3-version systems:

$$P_V = \frac{F_1}{NF_0 + F_1}, \tag{4}$$

$$P_{RV} = \frac{2F_2P_V(1-P_V) - (N-1)F_1P_V^2}{2F_2P_V(1-P_V) + (N-1)F_1(1-P_V^2)}, \tag{5}$$

$$P_{RALL} = \frac{F_3 - {P_V}^3}{1 - {P_V}^3}, \tag{6}$$

where F_1,F_2 and F_3 represent the observed frequency of a single failure, two and three coincident failures, respectively, in a 3-version configuration.

In order to verify the effectiveness and consistency of DL model, we apply new data to this model and compare our results with original results in [7]. In this experiment, we employ the same six mutants passing the qualification test as the target versions in this fault tree model and their failure characteristics are investigated. The 400 operational test cases were executed on these

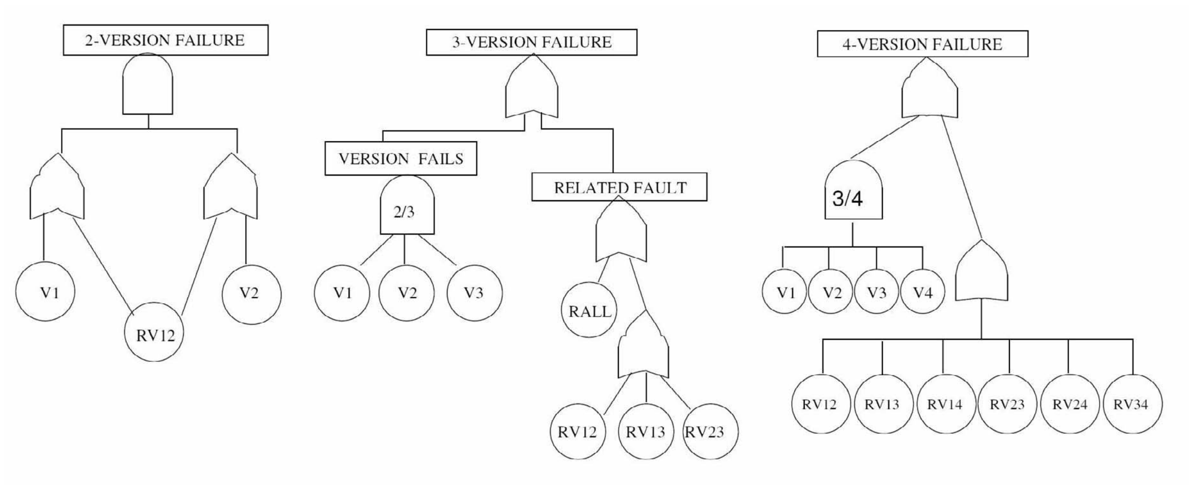

Figure 1. Fault tree models for 2, 3 and 4 version systems (from [7])

mutants and the failures encounter in each mutant are shown in Table 8. We can see from Table 8 that the average failure probability for single version is 0.01, which is much smaller compared with the original experimental data in [7]. It indicates that the versions we used in this experiment is more reliable. Moreover, the small failure frequency does not affect the prediction accuracy in terms of magnitude.

Table 8. failure Characteristics for individual mutants

Mutant ID	Number of failures	Prob. By-case
117	3	0.0075
215	1	0.0025
223	3	0.0075
305	5	0.0125
382	9	0.0225
403	3	0.0075
Average	4	0.01

We configure the six mutants in pairs, and compare their outputs for each test case. Table 9 yields an estimate of $P_V = 0.0084$ for the probability of raising an unrelated failure in a 2-version configuration, and an estimate $P_{RV} = 0.0016$ for the probability of a related failure.

Next, the six mutants are configured in sets of three. Table 10 shows the number of times that 0, 1, 2 and 3 failures occurred in the 3-version configuration. The data yields an estimate of $P_V = 0.0071$ for the probability of an independent failure. The comparison between the predicted probability of 0, 1, 2 and 3 failures using independence model and observed frequency are shown Table 11. Unlike the previous experiment reported in [6], our data shows that the observed frequency for two and three simultaneous failures is higher than that of the independence model. The data also yields the estimation of $P_{RV} = 0.0013$ for the probability of two related failures, and $P_{RALL} = 0.0004$ for the probability of failures involving all three versions.

Table 9. failure Characteristics for 2-version configurations

Category	Number of cases	Frequency
F0 - no failure	5890	0.9817
F1 - single failure	100	0.0167
F2 - two coincident	10	0.0017
Total	6000	1.0000

The mutants are then analyzed in combinations of four programs. Table 12 shows the number of times that 0, 1, 2, 3 and 4 failures occurring in the 4-version configuration. The data yields an estimate of $P_V = 0.0063$ for the probability of an independent failure. The comparison between the predicted probability of 0, 1, 2, 3 and 4 failures using independence model and observed frequency

Table 10. failure Characteristics for 3-version configurations

Category	Number of cases	Frequency
F0 - no failure	7797	0.9746
F1 - single failure	169	0.0211
F2 - two coincident	31	0.0039
F3 - three coincident	3	0.0004
Total	8000	1.0000

Table 11. Comparison of independent model with observed data for 3 versions

No. of failures	Independent model	Observed frequency
0	0.9786	0.9746
1	0.0213	0.0211
2	0.0002	0.0039
3	0	0.0004

are shown in Table 13. Just like 3-version configuration, our data shows that the observed frequency for three and four coincident failures is higher than that of the independence model. The data also yields the estimation of $P_{RV} = 0.0028$ for the probability of two related faults, and $P_{RALL} = 0$ for the probability of coincident failures in all four versions.

Table 12. failure Characteristics for 4-version configurations

Category	Number of cases	Frequency
F0 - no failure	5811	0.9685
F1 - single failure	147	0.0245
F2 - two coincident	33	0.0055
F3 - three coincident	9	0.0015
F4 - four coincident	0	0.0000
Total	6000	1.0000

Table 14 summarizes the parameters estimated from our data. The parameters are applied to the fault tree model shown in Figure 1. The predicted system failure probability using derived parameters in the fault tree models agrees quite well with the observed data, especially with the 2- and 3-version configurations. For the 4-version configuration, the predicted probability is close to zero but the observed frequency is 0.0015. Our experiment shows that the predicted system failure probability from fault tree model is very close to the observed values in most situations, except that there is a gap between the two in 4-version model. This should be further investigated to validate the effectiveness and accuracy of the fault tree model.

Table 13. Comparison of independent model with observed data for 4 versions

No. of failures	Independent model	Observed frequency
0	0.9750	0.9685
1	0.0247	0.0245
2	0.0002	0.0055
3	0	0.0015
4	0	0

Table 14. Summary of parameter values derived from our data

2-version model	3-version model	4-version model
$P_V = 0.0084$	P_V=0.0072	P_V=0.0063
$P_{RV} = 0.0016$	$P_{RV} = 0.0013$	P_{RV}=0.0028
	P_{RALL}= 0.0004	P_{RALL}= 0
Predicted system failure probability (from the model)		
0.0017	0.0045	0.000048
Predicted system failure probability (from the data)		
0.0017	0.0043	0.0015

Figure 2 compares the predicted reliability of three different configurations, including 2-version configuration for Distributed Recovery Block (DRB) [18], 3-version configuration for N-Version Programming (NVP) [2, 15], and 4-version configuration for N-Self Checking Programming (NSCP) [12]. We can see from our experiment that DRB is the most reliable of the three to produce a correct result, while NSCP is the least reliable. Compared with the original experimental data in [6], the prediction performance of the three configurations in our experiment are consistent with those in [6]. However, if we look into the first hundreds of hours, the three configurations performs differently, as shown Figure 3. Here NSCP depicts higher reliability than DRB and NVP, although it gives the least reliability in the long run.

Figure 4 compares the predicted safety of the three systems. Here we assume that the decider

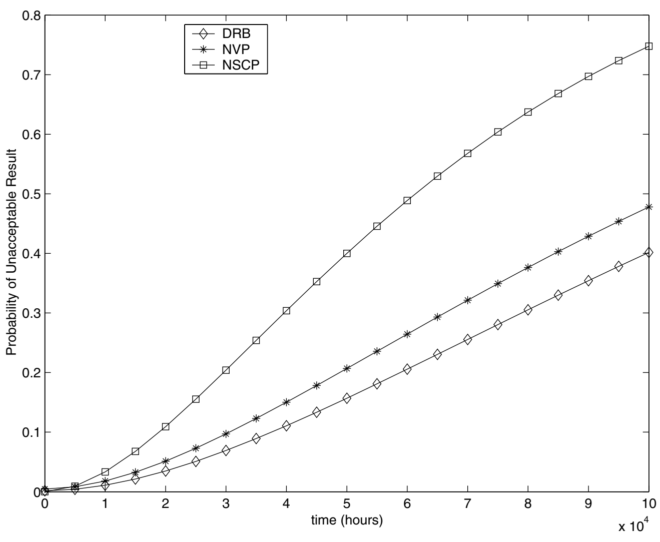

Figure 2. Predicted reliability by different configurations

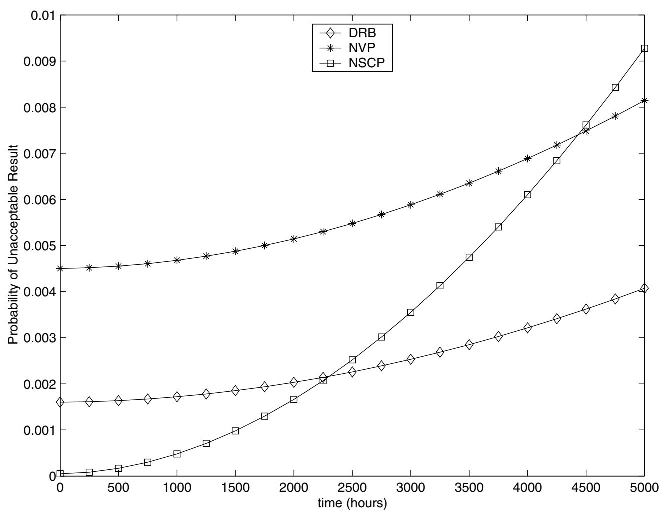

Figure 3. Predicted reliability by different configurations

used in the NVP and NSCP has a failure probability of only 0.0001 and that for DRB has a failure rate of 0.001 [7]. According to Figure 4, NSCP is the most likely to produce a safe result, while DRB are an order of magnitude less safe than NSCP. This is also consistent with the original experimental results in [7].

Overall, compared our project with former project in [7], the reliability and safety performance of DRB, NVP, NSCP shows consistency of DL model with respect to our experimental data. The discrepancy in the first hundreds of hours may indicate dependence on operational domains and needs further investigations. Furthermore, the above predictions are on the basis of our primary data, some assumptions in [7] and the fault tree modeling. To achieve more accurate results, the information about the correlation between successive executions should be included [19].

5 Conclusion and Future Work

In this paper, we perform analysis and investigation on reliability and fault correlation issues for diverse software systems. We apply our RSDIMU project data to evaluate the effectiveness and prediction accuracy of existing reliability models for fault tolerant software. In our experiment, mutants with real faults are engaged. 400 operational test cases were executed on six mutants which passed a qualification test to investigate the fault correlation features between any pairs of mutants.

We first apply Popov, Strigini et al's reliability bounds model to locate the upper and lower bounds for reliability of diverse programs. The results reveal that the confidence bounds are tighter with our data in most situations. It verifies the hypothesis of "variety of difficulties" on different demand subdomains, and supports the effectiveness of design diversity with small fraction of positive fault correlations and existence of negative correlations. Furthermore, we adopt Dugan and Lyu's dependability model to parameterize the reliability of different configurations. The analysis shows that NSCP is the least reliable but most safe approach among the three, while DRB inherits the highest reliabililty but the lowest safety according to our experimental data in the long run. The discrepancies in the first hundreds of hours may relate to the operational domain and needs further investigation.

As our future work, we will further analyze the prediction accuracy of these reliability and fault

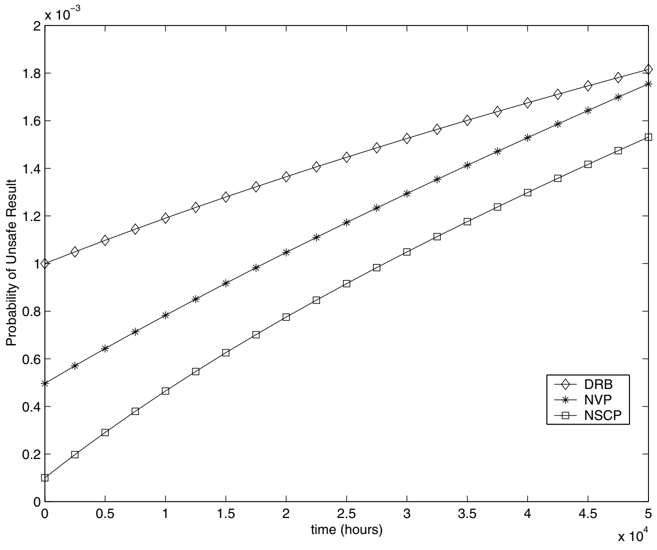

Figure 4. Predicted safety by different configurations

correlation models on design diversity. Other comprehensive models such as Tomek and Trivedi's model using stochastic reward nets will be parameterized and analyzed by our data set. Further testing and verification will be explored on our data set to collect more experimental results.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4360/02E).

The authors would like to greatly appreciate Lorenzo Strigini for his detailed review of this paper and his extensive comments leading to a great improvement of the paper in its technical content and research direction.

References

[1] J. Arlat, K. Kanoun, and J. C. Laprie. Dependability modeling and evaluation of software fault-tolerant systems. *IEEE Transactions on Computers*, 39(4):504–513, September 1990.

[2] A. Avizienis. The n-version approach to fault-tolerant software. *IEEE Transactions on Software Engineering*, 11(12):1491–1501, December 1985.

[3] P. G. Bishop. Software fault tolerance by design diversity. In M. R. Lyu, editor, *Software Fault Tolerance*. Wiley, New York, 1995.

[4] A. Bondavalli, S. Chiaradonna, F. D. Giandomenico, and L. Strigini. A contribution to the evaluation of the reliability of iterative-execution software. *Software Testing, Verification, and Reliability*, 9(3):145–166, March 1999.

[5] A. Csenki. Recovery block reliability analysis with failure clustering. In A. Avizienis and J. C. Laprie, editors, *Dependable Computing for Critical Applications(DCCA-1)*. Santa Barbara, USA, 1991.

[6] J. B. Dugan and M. R. Lyu. System reliability analysis of an n-version programming application. *IEEE Transactions on Reliability*, 43(4):513–519, December 1994.

[7] J. B. Dugan and M. R. Lyu. Dependability modeling for fault-tolerant software and systems. In M. R. Lyu, editor, *Software Fault Tolerance*. Wiley, New York, 1995.

[8] D. E. Eckhardt, Caglavan, Knight, Lee, McAllister, Vouk, and Kelly. An experimental evaluation of software redundancy as a strategy for improving reliability. *IEEE Transactions on Software Engineering*, 17(7):692–702, July 1991.

[9] 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*, 11(12):1511–1517, December 1985.

[10] A. Gnrarov, J. Arlat, and A. Avizienis. On the performance of software fault-tolerance strategies. In *Proceedings 10th International Symposium on Fault Tolerant Computing (FTCS-10)*, pages 251–253, Kyoto,Japan, July 1980.

[11] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, 12(1):96–109, January 1986.

[12] J. C. Laprie, J. Arlat, C. Beounes, and K. Kanoun. Definition and analysis of hardware- and software- fault-tolerant architectures. *IEEE Computer*, 23(7):39–51, July 1990.

[13] B. Littlewood and D. Miller. Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering*, 15(12):1596–1614, December 1989.

[14] M. R. Lyu. *Software Fault Tolerance*. Wiley, New York, 1995.

[15] M. R. Lyu and Y. He. Improving the n-version programming process through the evolution of a design paradigm. *IEEE Transactions on Reliability*, 42(2):179–189, June 1993.

[16] M. R. Lyu, Z. Huang, K. S. Sze, and X. Cai. An empirical study on testing and fault tolerance for software reliability engineering. In *Proceedings 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'2003)*, pages 119–130, Denver, Colorado, November 2003.

[17] P. T. Popov, L. Strigini, J. May, and S. Kuball. Estimating bounds on the reliability of diverse systems. *IEEE Transactions on Software Engineering*, 29(4):345–359, April 2003.

[18] B. Randell. System architecture for software fault tolerance. *IEEE Transaction on Software Engineering*, 7(6):220–232, June 1975.

[19] L. Strigini. On testing process control software for reliability assessment: the effects of correlation between successive failures. *Software Testing Verification and Reliability*, 6(1):33–48, January 1996.

[20] A. T. Tai, A. Avizienis, and J. F. Meyer. Performability enhancement of fault-tolerant software. *IEEE Transactions on Reliability, Special Issue on Fault-Tolerant Software*, 42(2):227–237, June 1993.

[21] L. A. Tomek and K. S. Trivedi. Analyses using stochastic reward nets. In M. R. Lyu, editor, *Software Fault Tolerance*. Wiley, New York, 1995.