Bishop, P. G. (2015). Modeling the Impact of Testing on Diverse Programs. Paper presented at the 34th International Conference, SAFECOMP 2015, 23-09-2015 - 25-09-2015, Delft, The Netherlands.

Modeling the Impact of Testing on Diverse Programs

Peter Bishop

City University and Adelard LLP, London, UK
pgb@csr.city.ac.uk, pgb@adelard.com

Abstract. This paper presents a model of diverse programs that assumes there are a common set of potential software faults that are more or less likely to exist in a specific program version. Testing is modeled as a specific ordering of the removal of faults from each program version. Different models of testing are examined where common and diverse test strategies are used for the diverse program versions. Under certain assumptions, theory suggests that a common test strategy could leave the proportion of common faults unchanged, while diverse test strategies are likely to reduce the proportion of common faults. A review of the available empirical evidence gives some support to the assumptions made in the fault-based model. We also consider how the proportion of common faults can be related to the expected reliability improvement.

Keywords: Software diversity, multi-version programs, diverse test strategies.

1 Introduction

When diverse programs are developed, different software versions are developed and debugged by independent teams. Diversity seeking decisions (DSDs) can be used both in the development approach (languages, software design, etc.) and in the verification and testing used to debug the software. This paper focuses primarily on the impact of using similar or diverse test strategies.

The impact that testing has on single and diverse programs has been examined before [5, 7] but those models were based on the standard "difficulty function" approach, and evaluated testing strategies in terms of their impact on the probability of failure on demand (*pfd*). This paper takes a related but different approach to assessing the impact of different test strategies where the model is expressed in terms of a set of "characteristic faults" that might be introduced into the software. The diversity of the program versions is modeled by the proportion of faults that are common between the versions. Test strategies are modeled by different orderings of bug removal in the two versions and the impact of different test strategies on program diversity are examined. The effectiveness of alternative test strategies in reducing the common faults proportion is modeled, and we discuss how the common fault proportion can be related to the expected improvement in reliability.

2 The Diverse Debug Model

Versions are developed and debugged by independent teams, and different approaches can be taken by both teams (whether by accident or design). So in general we need to consider the:

- Likelihood of inclusion of faults in development;
- Likelihood of removal by testing.

In this model we assume that:

1. There are "characteristic program faults", i.e. different developers can make identical mistakes. There is a finite pool of N possible faults available for selection;
2. Faults from this pool are independently selected by the two development teams, i.e. a given fault, i, is selected with probability $s(i)$ by both development teams;
3. The test strategies examined are equally effective at removing faults, i.e. remove the same proportion f of the faults present in the diverse programs.

2.1 Fault Inclusion Model

The fault selection probability $s(i)$ is assumed to be the same for both teams, so the probability that fault i is common to versions A and B is:

$$P_{AB}(i) = s(i)^2$$

This is similar to the Eckhardt and Lee difficulty function model [3] for modeling common mode failure where there is a probability that a given *input point* is faulty. The fault selection model could perhaps be generalized to have different selection probabilities for A and B (similar to the Littlewood and Miller difficulty model [4]), i.e.:

$$P_{AB}(i) = s_A(i)s_B(i)$$

However an assumption of common $s(i)$ values for both versions is the worst case, and will be taken to hold in the remainder of this paper.

If we now consider the whole set of N characteristic faults where $i = 1 \ldots N$, the mean number of single version faults is:

$$N_A = N_B = \sum s(i)$$

The mean number of common faults is:

$$N_{AB} = \sum s(i)^2$$

The mean proportion of common faults is:

$$\beta = \frac{N_{AB}}{N_A}$$

So substituting for N_A and N_{AB}:

$$\beta = \frac{\sum s(i)^2}{\sum s(i)} \tag{1}$$

2.2 Modeling Fault Removal

Testing changes the existence probability of the characteristic faults. We can model the impact of diverse testing by two test reduction probabilities, $t_A(i)$ and $t_B(i)$. The test reduction probability $t_A(i)$ is the probability that fault i will be removed after applying the test strategy used for version A. So the expected number of faults remaining in a program version is:

$$N'_A = \sum s(i)t_A(i)\,, \qquad N'_B = \sum s(i)t_B(i) \tag{2}$$

The number of faults remaining can differ for versions A and B, as different test strategies may be deployed by the two teams, but we have made an assumption (see assumption 3) that the teams will choose equally effective test strategies, where there is similar reduction f in the proportion of faults in both versions, i.e.:

$$N'_A = N'_B = fN_A = fN_B \tag{3}$$

So after testing, the expected proportion of common faults, β', is:

$$\beta' = \frac{N'_{AB}}{N'_A} = \frac{N'_{AB}}{N'_B} = \frac{\sum s(i)^2 t_A(i)t_B(i)}{\sum s(i)t_A(i)} = \frac{\sum s(i)^2 t_A(i)t_B(i)}{\sum s(i)t_B(i)} \tag{4}$$

We can use this equation to model the impact of different test strategies by assigning different probability distributions to $t_A(i)$, $t_B(i)$.

3 Modeling Different Test Strategies

We can use the model to examine combinations of some extreme test strategies for the two versions, namely:

- Random removal, where faults are randomly removed from each version;
- Strictly ordered removal, with the same removal sequence in both versions;
- Strictly ordered removal with a sequence that depends on selection probability.

These test strategies may not necessarily be realistic, but they can help identify the best and worst cases achievable by different test strategies.

3.1 Random Removal Test Strategy

In the random removal strategy we assume that, for all faults 1...*N*:

$$t_A(i) = t_B(i) = f$$

From equation (2), it follows that:

$$N'_A = \sum s(i) f \ , \qquad N'_{AB} = \sum (s(i) f)^2$$

So it follows the fault reduction factor is also *f*. From equation (4):

$$\beta' = \frac{\sum s(i)^2 f^2}{\sum s(i) f} = f \frac{\sum s(i)^2}{\sum s(i)} \tag{5}$$

From the original definition of β in equation (1):

$$\beta' = f\beta$$

As a result, the proportion of common faults decreases as the number of residual faults decreases, i.e.:

$$\beta' \to 0, \quad f \to 0$$

3.2 Ordered Removal Strategy (with Independence)

In an ordered removal test strategy all faults will be removed in a specific order in both versions. This would, for example, occur if both program versions were tested with a common test set. Without loss of generality we can assume that the removal order runs from *N* down to 1. With some fraction *f* of the *N* potential faults remaining:

$$t_A(j) = t_B(j) = 1, \qquad j \le Nf$$

$$t_A(k) = t_B(k) = 0, \qquad k > Nf$$

If we consider the case where the test effectiveness probability of defect $t(i)$ is independent of the defect inclusion probability $s(i)$, then:

$$\beta' = \frac{N'_{AB}}{N'_A} = \frac{\sum s(j)^2}{\sum s(j)} \approx \frac{f \sum s(i)^2}{f \sum s(i)} = \beta \tag{6}$$

Hence with exactly the same order of removal of common faults in the diverse programs, we expect the proportion of common defects to remain constant (on average) as testing proceeds, i.e.:

$$\beta' \approx \beta, \quad f \to 0$$

3.3 Ordered Plus Random Removal Strategies (with Independence)

In this scenario, one team follows a random removal strategy and the other team uses an ordered removal strategy and we assume the removal order is independent of the selection probability. Both tests reduce the number of faults by the same fraction f so only faults j=1… fN remain in one version and all faults in the other version have a survival probability f. It follows that the proportion of common faults after testing is:

$$\beta' = \frac{N'_{AB}}{N'_A} = \frac{N'_{AB}}{N'_B} = \frac{\sum s(j) \cdot s(j) f}{\sum s(j)} \approx \frac{f \sum s(i)^2}{\sum s(i)} \tag{7}$$

For this combination of test strategies, it follows that:

$$\beta' \approx f\beta$$

And in the limit:

$$\beta' \to 0, \quad f \to 0$$

3.4 Ordering Dependent on Inclusion Probability

If the defects are removed in sequence from j=N to j=1 and the removal order is strictly dependent on the inclusion probability then:

$$\max s(i) = s(j=1) \geq s(j=2) \geq \ldots \geq s(j=N) = \min s(i)$$

So rare faults will be detected and removed first and faults that occur frequently in software will be removed last. With this ordering:

$$\beta' = \frac{\sum_{j=1}^{\Delta N} s(j)^2}{\sum_{j=1}^{\Delta N} s(j)} \leq \frac{\Delta N \max s(i)^2}{\Delta N \max s(i)} \leq \max s(i), \qquad f = \frac{\sum_{j=1}^{\Delta N} s(j)}{\sum_{j=1}^{N} s(j)} \tag{8}$$

Where ΔN is the number of potential faults remaining. In the limit as $\Delta N \to 0$:

$$\beta' \to \max s(i), \qquad f \to 0$$

Conversely, the reverse removal order where commonly occurring faults are removed first will leave the rarely occurring faults until last. So in the best case:

$$\beta' \to \min s(i), \qquad f \to 0$$

3.5 Dependent Order Plus Random Removal Strategies

If one removal strategy is random with detection probability f and the other strategy is ordered with dependency between $s(i)$ and the removal order, it follows that the proportion of common faults is bounded by:

$$f \min s(i) \le \beta' \le f \max s(i)$$

Even in the worst case where $\beta' \le f \max s(i)$, it follows that:

$$\beta' \le \beta, \qquad f < \frac{\text{mean } s(i)}{\max s(i)}$$

So the proportion of common faults will still reduce when $f \to 0$.

4 Summary of the Model Results

The fault-based model results are summarized in **Table 1** below, where:

- f is the fraction of faults remaining after testing;
- β is the expected proportion of common faults before testing starts;
- β' is the expected proportion after testing.

Table 1. Common fault proportion after testing (different test assumptions)

Test strategy A	Test strategy B	Correlation with $s(i)$	Expected β'
ordered	same order	none	$\approx\beta$
random	random	none	$f\beta$
ordered	random	none	$\approx f\beta$
ordered	same order	small s removed first	$\to \max s(i)$
ordered	random	small s removed first	$\to f \max s(i)$
ordered	same order	large s removed first	$\to \min s(i)$
ordered	random	large s removed first	$\to f \min s(i)$

It can be seen that the overall proportion of common faults depends on the correlation between:

- The two test strategies;
- Each test strategy and the fault selection probability.

In practice, the test strategies are unlikely to correspond exactly with any of the specific cases shown in Table 1, but we can see that, even with the *same* test set, it is possible for β' to remain the same, and with differing test strategies, it is likely that the proportion of common faults will decrease.

This observation is however only valid if the model assumptions are valid. This will be considered in more detail in the next section.

5 Validity of Model Assumptions

5.1 Set of Characteristic Faults

There is some experimental evidence that programmers make similar mistakes that result in a set of characteristic faults. The "programming contest" programs [10] provide a useful research resource as there are many thousands of implementations of particular contest problems. Research studies [1, 11] have shown that many programmers can produce exactly the same characteristic faults.

On a smaller scale, but with more realistic programs, an analysis of faults found in the Knight and Leveson diversity experiment showed that similar faults existed in multiple versions [2] where (possibly) 7 out of 27 versions contained a similar fault.

There is therefore considerable support for the assumption that characteristic faults exist in programs (typically related to specific functions that are implemented by the program).

5.2 Independent Selection of Faults from the Pool

An analysis of the Knight and Leveson diversity experiment [2] found 45 faults in 27 versions. If we take these to be the characteristic set of N faults, we can use the data to test the independent selection assumption. With independent selection, the probability of a perfect version, p_p, is:

$$p_p = \prod_{i=1,N} (1 - s(i))$$

It should be noted that dependence in fault selection would actually *increase* the value of p_p. For example, if $s(1)=s(2)=0.5$, then $p_p=0.25$ but if the two faults were always selected together, then $p_p=0.5$.

In the Knight and Leveson example, the number of faults found was $N=45$, and the average faults per version is 1.67. With $s(i)$ assumed to be identical for all faults at $s=1.67/45=0.037$. Assuming independent selection, $p_p=(1-0.037)^{45}=0.183$, so $(27\times0.183)=4.95$ versions are expected to be perfect. This is close to the actual figure of 6 perfect versions out of 27. Given the sampling uncertainties involved, the two numbers are in good agreement, which suggests the independent selection assumption is a reasonable approximation in this example.

5.3 Same Fault Selection Probabilities by Diverse Teams

The fault selection probabilities could differ in the diverse teams (e.g. due to different development techniques). It might even be the case that some faults, like array bound violations, are impossible in some technologies, i.e. $s_A(i)=0$. In the most extreme case,

the selection probabilities could be completely disjoint (i.e. complete negative correlation between s_A and s_B). So the assumption of similar detection probabilities (i.e. complete positive correlation between s_A and s_B) is the most pessimistic assumption as it maximizes β.

5.4 Independence of Fault Removal Order and Fault Selection Probability

The credibility of independence between fault selection probability and fault removal is difficult to determine empirically, but we do have some experimental evidence for cases where the removal is ordered by the fault "size" (where fault size is defined as the proportion of input space occupied by the fault). Typically we would expect faults detected by dynamic testing to fail more often if the faults are larger, so the removal order would be related to fault size. With independence, we would expect similar inclusion probabilities regardless of fault size. Unpublished data from research undertaken for [11] is shown in **Fig. 1** below. This shows the distribution of failure region sizes as a proportion of the whole input space. It should be noted that an "equivalence class" can be a "basic fault" (where programmers make the same mistakes in different versions) or a combination of two or more "basic faults".

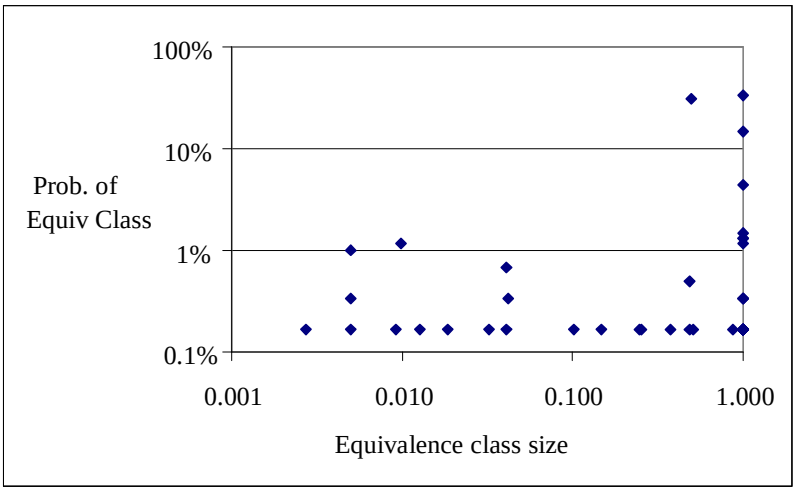

Fig. 1. Selection probability versus fault size

It can be seen that, apart from the very large equivalence class sizes, the inclusion probability varies between 0.1% and 1% for most classes. So with ordered removal we would expect β' to be 10^{-2} or less during fault removal.

A preliminary review of the distribution of failure rates in Knight and Leveson data (used as a surrogate for fault size) indicates that there is one case where 4 versions in 27 had faults with identical failure rates that were related to the same basic function. If this is due to the same characteristic fault i being present in 4 versions, then $s(i)$~15%. There are also 3 cases where faults in 2 versions out of 27 have similar

rates (i.e. $s(i)$~7.5%). The remaining 35 faults only appear in one version ($s(i)$~3.7%). These fault selection probabilities appeared to have little correlation with their associated failure rates.

These limited results suggest, at least in some cases, a presumption of independence between fault inclusion and removal may be credible. However more empirical studies are needed to determine whether such independence is likely for test strategies in general.

6 Discussion

6.1 Relationship to Prior Research

Littlewood et al. [5] showed that diverse test strategies can be more effective at improving reliability than a single test strategy, but only in the context of a single program version. Popov and Strigini [8] proposed a fault-based model for diverse programs for modeling the improvement achievable by diversity, but did not explicitly consider the impact of testing or diverse test strategies. Popov and Littlewood [7] considered the impact of a range of test strategies on diverse program versions during development, but the model is constructed at the level of specific points in the input space (rather than the failure regions covered by a fault) so it is difficult to represent fault inclusion and fault removal explicitly.

Unlike these earlier models, the model presented in this paper has a more limited scope as there is no attempt to estimate the individual and joint *pfd*s of the diverse versions. The analysis in this paper only sought to estimate the *proportion* of faults that are likely to be common (β) and how this proportion is affected by testing (β'). The relationship of β' to individual and joint *pfd*s is indirect. We would expect that a smaller β' would result in a smaller joint *pfd*, but this might not be true for all usage profiles.

For the model to be applicable, we need to be confident that the underlying assumptions are valid and that the β model is a useful measure of the potential reliability improvement. The evidence presented in Section 5 provides some justification that the fault inclusion assumptions are credible and shows how an analysis of the number of common faults observed during the testing of diverse versions could provide an empirical estimate for β.

In the remainder of this discussion we consider the relationship of β to the expected reliability improvement and how this might be justified. We also consider how the reduction in β predicted for diverse strategies could be validated experimentally.

6.2 Relationship of β to *pfd*

While our model indicates that the choice of testing strategies could either keep the proportion of common faults β constant or even decrease the proportion, it is difficult to relate the results directly to the expected improvement in reliability of a diverse

pair of programs under a given choice of test strategies. To make a link with *pfd*, further modeling assumptions need to be made.

If there are n faults in versions A and B, and $n\beta$ are common faults then:

$$pfd_A = \sum_{i=1,n\beta} p_{AB}(i) + \sum_{j=n\beta+1,n} p_A(j), \qquad pfd_B = \sum_{i=1,n\beta} p_{AB}(i) + \sum_{k=n\beta+1,n} p_B(k)$$

Where $p_{AB}(1..\beta n)$ are the probabilities of failure of the common faults, $p_A(\beta n+1..n)$ are the probabilities of failure of the unique faults in version A, and $p_B(\beta n+1..n)$ are the probabilities of failure of the unique faults in version B.

If we further assume that the failure probabilities are similar for common and unique faults, i.e. $Ep_{AB}(i) \approx Ep_A(j) \approx Ep_B(k)$, then the *pfd*s of A and B are similar, i.e.:

$$pfd_A \approx pfd_B = pfd$$

Where *pfd* is the average probability of failure on demand of a single version. It also follows that the probability of simultaneous failure due to the common faults is:

$$pfd_{common} \approx \beta \cdot pfd$$

If we further assume that the unique faults fail independently between versions, then the probability of coincident failure between versions due to the unique faults is:

$$pfd_{unique} = \sum_{j=n\beta+1,n} p_A(j) \cdot \sum_{k=n\beta+1,n} p_B(k) \approx ((1-\beta) \cdot pfd)^2$$

Combining these two contributions, the overall probability of coincident failure between the versions, pfd_{pair}, is:

$$pfd_{pair} = \beta \cdot pfd + (1-\beta)^2 pfd^2 \tag{9}$$

If we define a reliability improvement ratio R as pfd / pfd_{pair}, then from equation (9):

$$R = \frac{1}{\beta + (1-\beta)^2 \cdot pfd} \tag{10}$$

When *pfd* is small:

$$R \to \frac{1}{\beta}, \qquad pfd \to 0$$

While for a large *pfd*, the second term in equation (10) is dominant, i.e.

$$R \to \frac{1}{(1-\beta)^2\, pfd}, \qquad pfd \ \to 1$$

So we would expect the improvement ratio R to be close to the independence assumption for large *pfd* values, where $R \approx 1/pfd$, and be asymptotic to a plateau value of $R=1/\beta$ for small *pfd* values.

There is some empirical support for these predictions from experiments undertaken in [12] and [9] using a large number of program versions produced in a programming contest [10]. The experimental analysis progressively removed the versions with the highest *pfd*s and, for the remaining versions, calculated the mean *pfd* of all single programs and program pairs (pfd_{pair}). An example of the resultant reliability improvement R ratio is shown in **Fig. 2** below (similar graphs were obtained for the other program examples).

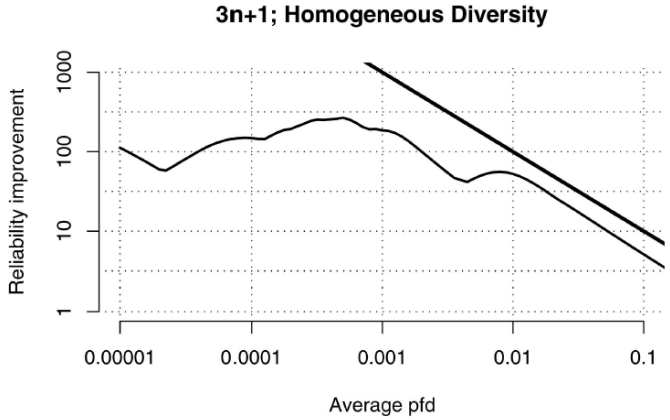

Fig. 2. Reliability improvement ratio vs. mean *pfd* of versions (van der Meulen et al. 2008)

It can be seen that for large mean *pfd* values, R is close to the reliability improvement expected when all failures are independent (the straight line). For small *pfd* values, the improvement reaches a plateau at around 100, which corresponds to a β value of around 0.01. Both these features are predicted by equation (9). The plateau at low *pfd* is expected for small *pfd* values where we expect R to remain constant at $1/\beta$. With strict removal ordering which is independent of inclusion probability $s(i)$, we would expect β' to be invariant, and we do see some evidence that a plateau has been reached. This would be consistent with inclusion probabilities $s(i)$ with a mean value of 0.01. The "ideal" behavior predicted by the model is shown in **Fig. 3** below (where $\beta=0.01$ is assumed).

The similarity of theory and experiment lends some support to the model proposed in equation (9). Note that this represents the improvement expected *on average*—not the actual improvement achieved for a specific program pair and usage profile.

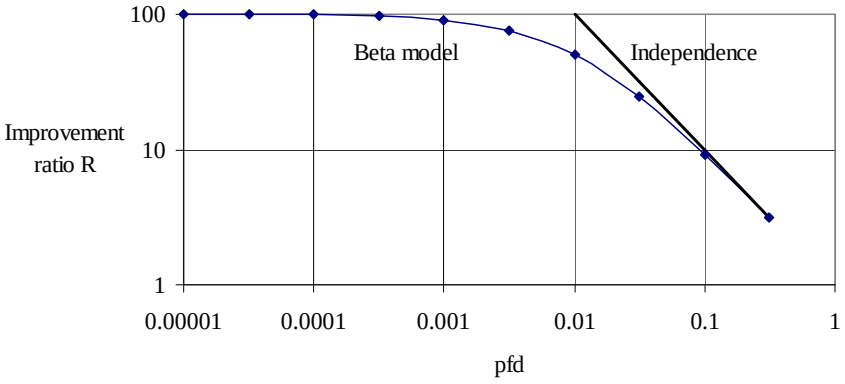

Fig. 3. Reliability improvement ratio (β model)

6.3 Validating the Performance of Diverse Test Strategies

One encouraging result from the model is that, given independence between fault inclusion probability and fault removal, a common test strategy does not necessarily increase the proportion of common faults, i.e. $\beta' = \beta$. However the model suggests that $\beta' < \beta$ is feasible if diverse test strategies are employed.

The impact of alternative strategies on β' could be evaluated empirically by simulating different fault removal strategies on the large set of program versions produced in programming contests such as [10].

7 Conclusions and Further Work

This paper has presented a model of diverse programs that assumes there are a common set of potential software faults that are more or less likely to exist in a specific program version.

Testing is modeled as a specific ordering of the removal of faults from each program version. Different models of testing were examined to derive the proportion of common faults as testing progresses.

Given the assumptions made, the theory suggests that a common test strategy (where common faults are removed in the same order in both program versions) could leave the proportion of common faults unchanged. So a common test strategy need not reduce the efficacy of diverse programs. We also show that diverse test strategies are likely to reduce the proportion of common faults. In the best case, where the test strategies are entirely uncorrelated, the common proportion after testing tends to zero.

We have also discussed how this proportion of common faults could be related to the expected reliability improvement.

Further work should be undertaken to:

- Validate the assumptions used in the model;
- Investigate the link between the proportion of common faults and the expected reliability improvement;
- Validate the testing strategy predictions in experimental research studies;
- Expand the theory to cover diverse development methods.

Acknowledgments. The author wishes to acknowledge the support of the UK Control and Instrumentation Nuclear Industry Forum (CINIF) who funded the research presented in this paper.

References

1. Bentley, J. G., Bishop, P. G., van der Meulen, M.: An empirical exploration of the difficulty function. In: International Conference on Computer Safety, Reliability, and Security, SAFECOMP 2004, LNCS, vol. 3219, pp. 60-71. Springer Berlin Heidelberg (2004)
2. Brilliant, S. S., Knight, J. C., Leveson, N. G.: Analysis of faults in an N-version software experiment. IEEE Transactions on Software Engineering 16(2), 238-247 (1990)
3. Eckhardt, D.E., Lee, L. D.: A theoretical basis for the analysis of multiversion software subject to coincident errors. IEEE Transactions on Software Engineering (12), 1511-1517 (1985)
4. Littlewood, B., Miller, D. R.: Conceptual modeling of coincident failures in multi-version software. IEEE Transactions on Software Engineering 15(12), 1596-1614 (1989)
5. Littlewood, B., Popov, P. T., Strigini, L., Shryane, N.: Modeling the effects of combining diverse software fault detection techniques. IEEE Transactions on Software Engineering 26(12), 1157-1167 (2000)
6. Littlewood, B., Rushby, J.: Reasoning about the Reliability of Diverse Two-Channel Systems in Which One Channel Is “Possibly Perfect”. IEEE Transactions on Software Engineering 38(5), 1178-1194 (2012)
7. Popov, P., Littlewood, B.: The effect of testing on reliability of fault-tolerant software. In: International Conference on Dependable Systems and Networks, DSN 2004, pp. 265-274. IEEE (2004)
8. Popov, P., Strigini, L.: The reliability of diverse systems: A contribution using modelling of the fault creation process. In: International Conference on Dependable Systems and Networks, DSN 2001, pp. 5-14). IEEE (2001)
9. Popov, P., Stankovic, V., Strigini, L.: An Empirical Study of the Effectiveness of “Forcing” Diversity Based on a Large Population of Diverse Programs. In International Conference on Software Reliability Engineering, ISSRE 2012, pp. 41-50. IEEE (2012)
10. Revilla, M. Skiena, S.: Programming Challenges: The Programming Contest Training Manual, Springer (2003)
11. Van der Meulen, M. J. P., Bishop, P. G., Villa, R.: An exploration of software faults and failure behaviour in a large population of programs. In: International Symposium on Software Reliability Engineering, ISSRE 2004, pp. 101-112. IEEE (2004)
12. Van der Meulen, M. J., Revilla, M. A.: The effectiveness of software diversity in a large population of programs. IEEE Transactions on Software Engineering 34(6), 753-764 (2008)