Towards Effective and Efficient High Order Mutation

Pedro Reales Mateo and Macario Polo Usaola

University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

Abstract. Mutation testing is a very effective testing technique. However, it remains very expensive. Several techniques to reduce costs have been proposed, One of them is High Order Mutation. This technique can reduce the costs of mutation testing with a decrement of the effectiveness. This paper proposes and evaluates a novel type of mutation, mutants-integration mutation that improves the effectiveness of High Order Mutation keeping its advantages. The result obtained in the experimentation leads one to think that using the new proposed type of mutation improves test cases generations when high order mutation is applied.

1 Introduction

Software testing is one of the most important tasks to ensure quality assurance. In academic studies several testing techniques have been designed to find different types of errors in different systems. One of the most effective testing techniques is mutation testing [1]. This technique is considered as a reference by the science community and it has been used traditionally to validate new proposals of testing techniques, although currently it is starting to be used in the industry.

With mutation testing, the tester has to create copies of the system under test with small syntactic changes (some of which represent errors) using mutation operators, which are well-formed rules. Then, the tester has to design test cases in order to find the inserted errors. Therefore, if the designed tests are able to find these simple errors, they will be able to find more complex ones due to the coupling effect [1].

One of the main problems of mutation testing is the cost. Researchers have put a lot of effort into the proposal of cost reduction techniques [2], being High Order Mutation (HOM) a very promising technique. With HOM, first order mutants (created by the injection of a single fault) are combined into high order mutants (created by the insertion of two or more faults, each proceeding from a first-order mutant). This reduces the number of mutants and therefore the costs of mutation testing. However, it also reduces the effectiveness of mutation testing because high order mutants are easily killed than first order mutants.

In this paper a new mutation type, named mutants-integration mutation, is proposed in order to keep the advantages of HOM and to reduce its disadvantages improving the effectiveness. This novel mutation type can be applied to high order mutants with the introduction of a more restricted condition to kill mutants that makes harder to kill mutants, so getting better quality tests cases.

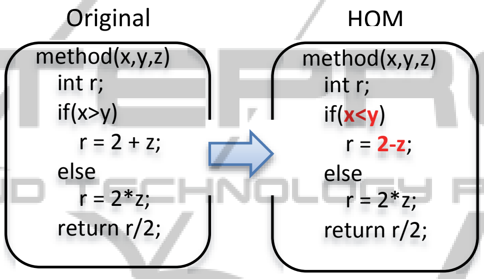
The paper is organized as follows. Section 2 describes High Order Mutation and

Reales Mateo P. and Polo Usaola M..
Towards Effective and Efficient High Order Mutation.
DOI: 10.5220/0004588200210029
In *Proceedings of the 1st International Workshop in Software Evolution and Modernization* (SEM-2013), pages 21-29
ISBN: 978-989-8565-66-2

its advantages and disadvantages. Section 3 presents mutants-integration mutation, the novel mutation type, which is the core of this paper. Section 4 explains the experimental setting performed in order to evaluate the work. Section 5 analyzes the results obtained in the experiment and, finally, section 6 shows the obtained conclusions and the future work derived from this study.

2 High Order Mutation

High order mutation [3] is an advanced testing technique. Using it, the tester introduces more than one fault in each copy of the system to create *high order mutants*. Figure 1 shows a 2nd-order mutant, where two mutations are combined (red text). In the literature there are two different approaches to apply high order mutation.



```
method(x,y,z)
  int r;
  if(x>y)
    r = 2 + z;
  else
    r = 2*z;
  return r/2;
```

```
method(x,y,z)
  int r;
  if(x<y)
    r = 2-z;
  else
    r = 2*z;
  return r/2;
```

Fig. 1. Example of a high order mutant.

The first approach [3], [4], tries to improve the efficiency of mutation testing using high order mutation. With this approach, first order mutants or traditional mutants are combined in order to reduce the total number of mutants. This reduction has two important advantages. First, the number of equivalent mutants is reduced from 20% of mutants to 5% [5], since most equivalent mutants are combined with non equivalent, so making non-equivalent high order mutants; and, second, the total number of mutants is reduced almost to half (in the case of 2nd-order mutation) [3], so the computational requirements decrease. However, this approach has a disadvantage: the effectiveness of mutation testing decreases [3], since some "good" mutants can be combined with "bad" mutants creating a bad high order mutant: a good mutant contains a difficult to reveal fault; if the HOM dies, it is probably because test cases have discovered the easy fault.

The second approach [6] tries to improve the effectiveness looking for HOMs in the search space of all the possible combinations of mutations. The goal is to find those HOMs that are more difficult to be killed than the FOM that composed them. The main advantage of this approach is that, since the final set of HOMs is more difficult to be killed, test cases designed for those mutants will have more quality than test cases designed with traditional mutants. However, the main disadvantage of this approach is related with its requirements: the generation of so good test cases needs very high computational requirements. since a search-based technique must be used to find the good high order mutants, what supposes several executions.

In order to overcome the problems of these two approaches and use HOM to improve the efficiency and effectiveness of mutation testing, a novel high order mutation technique is proposed in this paper. The main novel contributions of this technique is a new mutation type, called mutants-integration mutation, that applies some special conditions that must be fulfilled before considering a mutant is killed. Next section describes this new kind of mutation in detail.

3 Efficient and Effective High Order Mutation

Theoretically, the combined effect of some mutations can produce some types of HOM that are more difficult to kill than the isolated mutants because of the masking or interactions effects [6]. However, sometimes the effect of the mutations of a HOM is not combined. For instance, two mutations in different branches of an if-then-else structure. In this case the high order mutant can be killed but only because the effect of one mutation, so the other mutation goes unnoticed.

3.1 Mutants-integration Mutation

Using a combination algorithm, like anyone of the proposed algorithms [3], the probability to create a high order mutant where the mutations combine their effects is low, and there are not warranties that mutations do not go unnoticed since the HOM can only be killed by the effect of one mutation.

In order to mitigate this risk (mutations go unnoticed) and make high order mutation testing more effective, a novel kind of mutation, called mutants-integration mutation, is proposed. Unlike other mutation types, like Strong mutation [1] or flexible weak mutation [7], a special condition is added to the output state of the system under test A new rule (equation 1) is applied to determine when a mutant is considered killed.

$$\begin{gathered} Kill(tc, hom) = output(tc, \mathrm{hom}) \neq output(tc, original) \\ AND \\ \forall\, fom \in hom\colon covers(tc, fom) = true \end{gathered} \tag{1}$$

The formula determines that a HOM is only considered killed when the output state of the mutant at the end of the execution is different from the original and all the FOMs that compose the HOM have been reached by the test case. This special condition ensures that at least all the mutations are executed in the same test execution path. This rule implies that a test that does not cover all the mutants in a HOM is not able to kill it, independently of the results of the test. Figure 2 illustrates when a high order mutant is considered killed under mutants-integration mutation.

Figure 2 shows that test *a* reaches all the mutations and its output is different from the output that would be obtained with the original system. In this case, all the required conditions are fulfilled and, therefore, *a* kills the mutant. However, *b* only reaches one of the two mutations that compose the HOM. Thus, although the outputs of test *b* are different when executed against the original and against the mutant, it cannot be considered as killed because not all the required conditions are fulfilled.

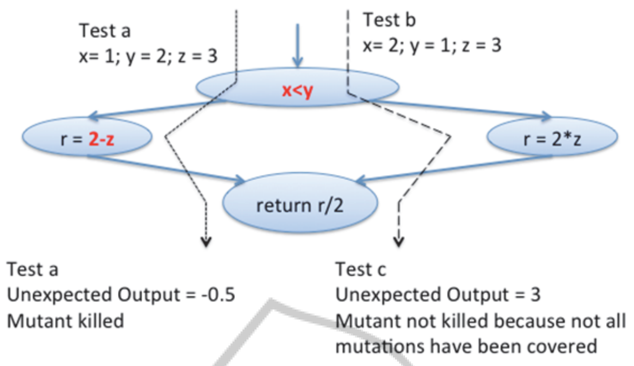


Fig. 2. An example of a killed mutant under mutants-integration mutation.

One of the main advantages of this new mutation type is related with the equivalent mutants. In previous work [5], when the set of total mutants is reduced using HOM, the number of equivalent mutants is also reduced. The required conditions by the new type of mutation implies that when an equivalent mutation is included in a high order mutant, the test only requires to cover it (jointly with the other mutations) and creates an unexpected output (produced by the other mutations) in order to kill the high order mutant. Therefore, the same advantage obtained in previous work remains.

However, there is not warranty that mutations do not go unnoticed, because it only requires that they are covered in the same execution path. This new mutation kind can increase the probability that mutations do not remain unnoticed.

4 Empirical Analysis

In order to investigate how the new mutation type improves the effectiveness of high order mutation, an experiment was performed. In this experiment we create some different sets of high order mutants with three different combinations algorithms proposed previously [3] and we generate test cases for each mutant set using strong mutation and mutants-integration mutation. Finally, we compare the quality of the generated tests executing them against the first order mutants. The next sections described the performed experiment.

4.1 Research Question

The main goal of this study is to determine if mutants-integration mutation is more effective than strong mutation when high order mutation is applied. Therefore the research question is established as:

Is Mutants-integration mutation more effective than strong mutation when high order mutation is used?

In order to evaluate this question, two variables were measured in the experiment. The independent variable is the mutation type used to create test case and has two values: strong mutation and mutants-integration mutation; the dependent variable is the mutation score achieved by the created test cases against the first order mutants.

4.2 Experimental Procedure

This section describes the experimental procedure, which is composed by 5 steps:

1- In the first step, first order mutants are created. These mutants will be combined to create 2nd-order mutants and will be used later to obtain comparable mutation scores.
2- In step 2, three combination algorithms (see section 4.5) are applied with the created first order mutants to create three sets of 2nd-order mutants.
3- Then, tests cases are generated automatically (see section 4.4) using strong mutation for each set of mutants.
4- In step 4, again test cases are generated automatically, but using mutants-integration mutation for each set of mutants.
5- Finally, when all tests are created, each test suite is executed again the first order mutants in order to obtain comparable mutation scores that will determine if mutants-integration mutation improves the effectiveness of high order mutation.

Therefore, at the end of the experimental process, there will be available thirteen different mutation scores for each selected class (see section 4.3) that can be compared (7 mutation scores obtained with the 1st-order mutants and 6 mutation scores obtained with the 2nd-order mutants).

4.3 Classes under Test

Two classes written in Java were selected for the presented experiments:

- *Board*. This class is from the monopoly application that has been used in previous studies [3]. This class implements the board of a monopoly game.
- *Month*. This class is from a medical appointment manager application used in the internal medicine consultation service at Hospital *Gutierrez Ortega*. This class implements a month in the system.

Table 1 shows quantitative information about the selected classes. The selection of these classes is motivated partially because the used test generation algorithm works properly with them.

Table 1. Classes under test.

Class	LOC	Methods	Mutants
Board	281	30	295
Month	478	18	708

4.4 Test Case Generation

One important element of the presented experiment is the test case generation. To generate automatically test cases, a genetic algorithm was used [8]. This algorithm is based on mutation testing and it is specially designed for Java software. It generates tests iteratively and selects the best tests based on the killed and covered mutants. Also, the algorithm implements a stop condition based on the evolution of the population (in this case, the generated test cases).

The same configuration of the algorithm was applied to generate tests each time. Table 3 of section 5 shows the number of generated tests for each set of mutants and each kind of mutation.

4.5 Combination Algorithms

In the experiment, three combination algorithms were used to create high order mutants: 1) *Each choice*, where first order mutants are combined in its order of generation and at least one time without any other restriction; 2) *Each choice first-last*, where first order mutants are combined in its inverse order of generation and at least one time without any other restriction; and 3) *Between operators*, where each first order mutant is combined at least one time with mutants created with different mutation operators. Table 2 shows the number of 2nd-order mutants obtained with each combination algorithm for each application.

Table 2. Number of 2nd-order mutants.

Class	Each Choice	Each choice first-last	Between operators
Month	354	354	459
Board	148	148	150

5 Results

This sections shows the results obtained from the experiment. Table 3 presents the tests cases generated and the mutation scores obtained with each test suite for the two classes included in this study.

5.1 Generated Test Cases

The generated test cases for each set of high order mutants are shown in table 3. It shows that when SM is used, the number of tests generated with 2nd-order mutants is lower than the number of tests generated with 1st-order mutants. This indicates, like in previous studies [3], that effectiveness of high order mutation is lower than first order mutation because mutants are easily killed. However, when MIM is used, the number of tests generated with the second order mutants is always higher than with the first order mutants. This issue indicates that it is more difficult to find tests cases that kill several mutants at the same time and therefore it is necessary to create more tests, which kill concrete mutants.

This data indicates that using MIM it is more difficult to kill a HOMthan using SM and, therefore, with MIM more test cases are generated that kills concrete mutants (under MIM).

Table 3. Experimental result.

	Test with 1stOrder Mutants	Test with 2ndOrder mutants BO		Test with 2ndOrder mutants EC		Test with 2ndOrder mutants FL	
	Kind of mutation						
	SM	SM	MIM	SM	MIM	SM	MIM
Month	**Number of tests**						
	129	81	238	78	174	78	144
	Mutation score against 1st-order Mutants (using SM)						
	82,61%	76,58%	78,73%	74,13%	76,72%	75%	77,58%
	Mutation score against 2nd-order Mutants						
	-	95,88%	82,9%	95,11%	80,17%	95,42%	78,57%
Board	**Number of tests**						
	58	49	93	38	64	40	52
	Mutation score against 1st-order Mutants (using SM)						
	87,79%	71,18%	73,89%	80%	81,69%	78,64%	71,18%
	Mutation score against 2nd-order Mutants						
	-	96%	72,66%	100%	63,51%	98,64%	52,02%

5.2 Mutation Scores against Second Order Mutants

Regarding to the mutation scores obtained against second order mutants (table 3), it shows that the score obtained under SM is always bigger that the score obtained under MIM. Again, this indicates that under MIM, it is more difficult to kill mutants (due to the special restrictions of MIM), and therefore, it is more difficult to reach a high mutation score.

Taking into account that the test cases were automatically generated with a genetic algorithm [8] using the same configuration in each case, it is possible to determine that the test design becomes more specific under MIM since more test cases must be generated and they kill less mutants that SM.

However, this issue introduces more costs to mutation testing, since test cases that reach a higher mutation score are more difficult to be designed, currently, with the automatic test generation techniques proposed in the literature [9], this cost increment is not really important and can be overcome automatically.

On the other hand, the combination of HOM and MIM mutation does not increment the number of equivalent mutants, therefore MIM keeps the cost reduction of HOM related with equivalent mutants [5]. Thus, under MIM, mutation scores with second order mutants similar than the mutation scores with second order mutants obtained under SM could be obtained, since there is the same number of equivalent mutants. Only, more test cases must be created.

5.3 Scores against First Order Mutants

Table 3 also shows the mutation score obtained by the test cases against the first order mutants. In all the cases (excepting the tests generated with the "BO mutants" of the Board system), the score obtained with the tests generated under MIM and 2nd-order mutants is higher than the score obtained with the test generated under SM and 2nd-order mutants. This result suggests that, in fact, the test cases obtained with MIM have better quality than test cases obtained with SM.

If we compare the mutation scores of the test cases obtained with MIM against the 1st and the 2nd-order mutants, it shows that the differences are low, around 1% and 4 % (only in two cases, "FL and BO mutants" of Board class, the difference is around 20%). And, moreover, these differences are not always negative. In some cases (for the board class) the mutation score obtained with second order mutants is lower than the mutation score obtained with first order mutants (which indicates that the high order mutants are more difficult to be killed than first order mutants).

In fact, the experiment results show that when a set of test executed against 2nd-order mutants under MIM reaches a mutation score lower than 75%, the same set of tests executed against a 1st-order mutants reaches a mutation score higher than the obtained against the 2nd-order mutants.

Finally, if we compare the mutation score obtained against 1st-order mutants by the test cases generated with MIM and by the tests cases generated directly with the first order mutants, it shows that the mutation score obtained by the test cases generated with the 1st-order mutants is always higher than the score obtained by the test cases generated with 2nd-mutants and MIM. This shows the drawback of MIM mutation: it reduces the probability that some mutations go unnoticed. However, as previously commented, more test cases could be generated under MIM in this experiment until they reach the same mutation score than the test cases obtained with 2nd-order mutants and SM, which will improved the mutation score obtained against first order mutants.

Summarizing, the results obtained from the experiment presented in table 3 shows that in general the test cases designed under MIM has higher quality than test cases designed under SM, and therefore, MIM reduces the drawbacks of high order mutation keeping its advantages. There, the question formulated in section 4.1, "*Is Mutants-integration mutation more effective than strong mutation when high order mutation is used?*" can be answered as "*yes, it is*".

5.4 Threads to Validity

There are some threads to validity that must be taken into account before accepting the obtained results. In experimentation there are three kinds of threads to validity [10]. First, the *construct validity,* which is the degree to which the variables are properly measured. In the experiment the only variable measured was the mutation score, which was properly measured, since tools measured it automatically.

Second, the *internal validity* is the degree of confidence in the cause-effect relationship between a factor and the experimental results. In the experiment, we observed than in the 83% of the cases the interesting factor (mutation type: SM and MIM) has the same effect. Thus, it must be taking into account than in the 17% of the cases the effect was different before to accept the results of the experiment presented in this paper.

Finally, the *external validity* is the degree to which the research results can be generalized. This is the most important weakness of the presented experiment, since only two classes were used and only one execution of the test generation algorithm was performed (one execution for each case). These issues must be taken into account before accepting the experimental results of this paper as definitive.

6 Conclusions

This paper presents a novel mutation type, mutants-integration mutation that increases the effectiveness of high order mutation and keeps its advantages. This mutation technique introduces a new restriction to consider that a mutant is killed, which produces that the designed tests have better quality than tests designed with traditional mutation.

The experimentation section of this paper shows some promising early results that leads to think that the theoretical advantages of mutants-integration mutation are fulfilled in practice. Moreover, these preliminary results show that with mutants-integration mutation and 2nd-order mutation it is possible to design test cases that achieve a mutation score similar against 2nd-order mutants and 1st-order mutants.

As future work, we plan to extend the empirical analysis of mutants-integration mutation in order to provide enough empirical data to validate the benefits of it. Also, we are defining a new algorithm to create second order mutants that complements mutants-integration mutation in order to obtain better results. Finally, we plan to develop a test generation algorithm specially designed for mutants-integration mutation, since, as section 5 shows, the current approach used in this paper cannot produce tests that reach very high mutation scores.

References

1. R. A. DeMillo, R. J. Lipton, and F. G. Sayward, “Hints on Test Data Selection: Help for the Practicing Programmer,” Computer, vol. 11, no. 4, pp. 34–41, Apr. 1978.
2. M. Polo and P. Reales, “Mutation Testing Cost Redution Techniques: A Survey,” IEEE Software, vol. 27, no. 3, pp. 80–86, Jun. 2010.
3. P. R. Mateo, M. P. Usaola, and J. L. F. Alemán, “Validating 2nd-Order Mutation at System Level,” IEEE Transactions on Software Engineering, vol. 99, no. 1, 5555.
4. M. Papadakis and N. Malevris, “An Empirical Evaluation of the First and Second Order Mutation Testing Strategies,” presented at the Software Testing, Verification, and Validation Workshops (ICSTW), 6, pp. 90–99.
5. M. Polo, M. Piattini, and I. García-Rodríguez, “Decreasing the cost of mutation testing with 2-order mutants,” Software Testing, Verification and Reliability, vol. 19, no. 2, pp. 111–131, 2008.
6. M. Harman, Y. Jia, and W. B. Langdon, “A Manifesto for Higher Order Mutation Testing,” in Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, Washington, DC, USA, 2010, pp. 80–89.
7. P. R. Mateo, M. P. Usaola, and J. Offutt, “Mutation at the multi-class and system levels,” Science of Computer Programming, 23.
8. Macario Polo and Pedro Reales, “Automated tests generation for multi-state systems,” presented at the Genetic and evolutionary computation conference, Amsterdam, 2013.
9. G. Fraser and A. Zeller, “Mutation-driven generation of unit tests and oracles,” 2010, p. 147.
10. Experimentation in software engineering: an introduction. Boston: Kluwer Academic, 2000.