SOFTWARE TESTING, VERIFICATION AND RELIABILITY
Softw. Test. Verif. Reliab. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/stvr.1529

Employing second-order mutation for isolating first-order equivalent mutants

Marinos Kintis[1,*,†], Mike Papadakis[2] and Nicos Malevris[1]

[1]*Athens University of Economics and Business, Athens, Greece*
[2]*Interdisciplinary Center for Security, Reliability and Trust, Luxembourg University, Walferdange, Luxembourg*

SUMMARY

The equivalent mutant problem is a major hindrance to mutation testing. Being undecidable in general, it is only susceptible to partial solutions. In this paper, mutant classification is utilised for isolating likely to be first-order equivalent mutants. A new classification technique, *Isolating Equivalent Mutants (I-EQM)*, is introduced and empirically investigated. The proposed approach employs a dynamic execution scheme that integrates the *impact* on the program execution of first-order mutants with the impact on the output of second-order mutants. An experimental study, conducted using two independently created sets of manually classified mutants selected from real-world programs revalidates previously published results and provides evidence for the effectiveness of the proposed technique. Overall, the study shows that I-EQM substantially improves previous methods by retrieving a considerably higher number of killable mutants, thus, amplifying the quality of the testing process. Copyright © 2014 John Wiley & Sons, Ltd.

Received 3 September 2012; Revised 24 October 2013; Accepted 19 March 2014

KEY WORDS: mutation testing; equivalent mutants; higher-order mutation; mutants' impact; dynamic analysis

1. INTRODUCTION

Among the various software development phases, testing usually constitutes more than half of the cost of the overall process. Hence, researchers are motivated to automate all the software testing activities in order to reduce its expenses. Unfortunately, this is not possible for all software testing tasks. Because testing involves many 'undecidable' problems [1], its full automation is impossible. Therefore, developing effective heuristics and achieving higher levels of automation for such problems is highly desirable.

Generally, the testing activity is performed by employing a set of test cases based on which the software's behaviour is explored. Thus, the testing process attempts to find an adequate test set that forces the software to exhibit a wide range of possible behaviours. In practice, this task is accomplished by the test adequacy criteria. Testing criteria impose a set of requirements to be exercised by the selected tests. These requirements offer practical solutions to the adequacy of testing, that is, when the testing process is stopped on the one hand, and guidance for constructing new tests is provided on the other.

Mutation testing involves the introduction of syntactic changes, called *mutants*, to the *original* program under test. Thus, multiple versions, called *mutant programs*, are produced that are syntactically different from the original program. Mutants are also termed according to their *order,* that is, the number of syntactic changes made to the original program. The term *higher-order* mutants is used to express mutants belonging to any order higher than one. The term *killed* is used to refer

*Correspondence to: Marinos Kintis, Athens University of Economics and Business, Athens, Greece.
†E-mail: kintism@aueb.gr

to a mutant that produces different output than the original program when they are both executed with the same test case. The term *live* refers to the opposite situation. The criterion requires the production of test cases that are capable of killing all generated mutants. Unfortunately, a significant number of mutants, termed equivalent mutants, cannot be killed because they are functionally equivalent to the original program.

Mutation testing is considered to be a rather powerful method, able to subsume or prob-subsume [2] most of the structural testing criteria [3, 4]. Additionally, empirical studies [5] suggest that mutation reveals more faults than most of the structural testing criteria. In the studies made by Andrews *et al.*, it has been shown that mutants exhibit a similar behaviour to real faults [6, 7]. Thus, if mutants can be killed, then programming faults might also be exposed. Although powerful, mutation is not widely used for testing. This may be because of its excessive computational demands, that is, mutation testing requires a huge number of mutants to be executed with test cases. To circumvent this problem, researchers have suggested various mutation approximation techniques [1, 8, 9] such as selective mutation [1, 8, 10] and mutant sampling [1, 8, 9]. These techniques aim at producing small mutant sets with similar power to the initial ones. The use of mutation approximation approaches is essential in order to make mutation scalable. The Javalanche framework [11], utilised in the present experimental evaluation, employs most of the mutation cost reduction advances.

Equivalent mutant instances introduce further difficulties to the mutation testing process. Their identification is essential in order to measure the adequacy of the performed testing activity. Adequacy, with respect to mutation testing, is measured by the ratio of the number of killed mutants to the entire number of introduced ones upon its reduction by the number of equivalent mutants. This adequacy measure is called mutation score. In order to be practical, the mutation score should be calculated automatically. In view of this, mutation should employ automated tools for generating and executing test cases with mutants and suggesting the adequacy of the performed process. Of the aforementioned problems, the automated production of test cases and the adequacy evaluation, that is, the elimination of equivalent mutants are difficult to handle 'undecidable' problems [1, 12, 13].

Recent advances in the area [14–18] achieved to automate the test cases production quite effectively. However, equivalent mutants' identification still lacks such techniques. In an attempt to overcome the difficulties caused by equivalent mutants, Schuler and Zeller [19, 20] suggested a technique, hereafter referred to as the *coverage impact* method, able to isolate possible equivalent mutants. This method is based on the observation that mutants affecting the program execution, are more likely to be killable than those that do not do so. Realizing this observation, mutants can be classified to those that affect the program's execution and those that do not. In the study of Schuler and Zeller, the first category comprised 75% of killable mutants. Thus, they concluded that practitioners should target this mutant set in order to restrict the side effects caused by equivalent mutants. Although, such a practice provides a highly likely to be killable mutant set, it was empirically found to consist of 56% of the total killable mutants, thus, missing a high number of killable mutants out. The approach described by the present paper expands the coverage impact method by aiming at overcoming the aforementioned limitation.

The proposed technique uses second-order mutants in order to isolate possible first-order equivalent mutants. The underlying idea is to determine the impact of a first-order mutant on other mutants. Thus, it is argued that a killable mutant is likely to impact the output of another mutant when both mutants are put together, that is, forming a second-order one. Although, such a technique is not clearly competitive to the *coverage impact* method, it enables the correct classification of different mutants. This fact motivated the design of a combined approach that utilises both aforementioned techniques with promising results.

The present work describes three variations of the proposed technique and investigates their ability to classify killable mutants. Additionally, it empirically validates the coverage impact method and compares it with the proposed ones. Experimental evaluation, conducted on two independently selected sets of manually classified mutants, confirms the coverage impact method's published results and reveals that its combination with the proposed approaches is capable of

DOI: 10.1002/stvr

retrieving approximately 20% more killable mutants. In general, the present paper contributes the following:

- A novel dynamic method to effectively classify first-order mutants using second-order ones.
- A case study validating the coverage impact method and the proposed approaches. In particular, the coverage impact method achieves a classification precision of 73% and a classification recall of 65%, whereas the proposed technique realises a precision score of 71% and a recall value of 81%, respectively.
- A manually identified set of killable and equivalent mutants, which amplifies an existing benchmark set [19, 20] and is made publicly available[‡] with the aim of enabling replication and comparative studies.

The rest of the paper is organised as follows: Sections 2 and 3 introduce the core concepts and describe the studied mutant classification techniques. Sections 4 and 5 present the utilised benchmarks and the experimental results. Sections 6 and 7 discuss issues regarding the mutants' impact and possible threats to validity. Section 8 refers to related work and in Section 9 conclusions and possible future directions are given.

2. MUTATION TESTING

This section introduces the equivalent mutant problem and describes the underlying concepts of the examined techniques. First, a description of the problems caused by the existence of equivalent mutants is given. Next, the concepts of first and higher-order mutation are discussed. The concept of mutants' impact is subsequently introduced. Finally, a mutation testing tool that is employed by the present study, named 'Javalanche', is described.

2.1. Equivalent mutants

Equivalent mutants introduce various difficulties in the mutation testing process. They are analogous to the infeasible elements of structural testing [12, 16]. However, their side effects are more serious. This is due to their number, which is high compared with the structural infeasible elements. Generally, the mutation testing process is composed of the following steps: (a) mutant creation; (b) test case generation; (c) mutant execution with the produced test cases; (d) elimination of the ineffective test cases (i.e., tests that do not kill any additional mutant); and (e) mutation score evaluation. This process is repeated iteratively (steps b, c and d), by producing and applying additional tests until reaching a predefined score threshold.

Every step of this process is influenced by the presence of equivalent mutants. Indeed, such mutants will be created without contributing to the testing process in step a. With respect to step b, the tester will spend effort aiming at killing those non killable mutants. With respect to step c, these mutants will be executed with all the test cases wasting valuable computational resources. Finally and more importantly, the mutation score is unknown in step e. Removing the equivalent mutants is hence crucial to complete this last step with confidence.

Ideally, equivalent mutants should be eliminated during step a. To do so, the knowledge of their equivalence is required. However, determining program equivalence, such as the equivalent mutant identification, has been shown, in general, to be an 'undecidable' problem [13]. Additionally, manually identifying one equivalent mutant requires approximately 15 min [19, 20]. Because mutation introduces a vast number of mutants, manual identification is usually impossible. Therefore, the present paper examines dynamic heuristics dealing with this problem. The employed heuristics use the mutants' impact [19–22] as a way to classify the live mutants as likely killable and likely equivalent ones. In other words, it provides an automated method to decide whether mutants can be killed or not. Being dynamic, the examined techniques necessitate information from mutant execution. Thus, they can be applied during step c. Application details of this approach are given in Section 3.

[‡]This set and the results of the present study are publicly available at http://pages.cs.aueb.gr/∼kintism/#stvr2014.

DOI: 10.1002/stvr

2.2. First-order and higher-order mutation

Mutation testing, also called mutation analysis, is a fault-based adequacy criterion. It operates by introducing mutants into the program's code and by comparing the differences they produce to the original program's output. Mutants are constructed by employing simple syntactic rules, termed *mutant operators*. By making one syntactic change at a time, *first-order mutants* (*foms*) are produced. By making *two*, *three* or *n* changes, *second* (*soms*), *third* (*toms*) and *n order* or generally higher-order (*homs*) mutants are formed. The production of differences in the program's output by the introduced mutants establishes the criterion requirements. Conversely, the tester's aim stemming from the criterion requirements is to produce tests capable of killing all introduced mutants.

Considering higher-order mutants has long been identified as an issue of the mutation analysis research. DeMillo *et al.* [23] proposed the coupling effect as 'Test data that distinguishes all programs differing from a correct one by only simple errors is so sensitive that it also implicitly distinguishes more complex errors'. This definition was later extended by Offutt [24] as the Mutation Coupling Effect Hypothesis, where first-order mutants were defined as simple faults, whereas higher-order as complex. In view of this, empirical evidence was provided and showed that tests able to kill first-order mutants are also capable of killing over a 99% of second and third-order ones [24]. As a consequence, the considered mutants were limited only to first-order ones.

Recently, Jia and Harman [25, 26] suggested using higher-order mutation as a possible answer to the difficulties faced by mutation testing. According to their study, there are a few but extremely valuable homs, termed 'subsuming' homs. These mutants are harder to kill than most of the foms and thus, one should aim at them only, ignoring most of the foms. In other studies, it was shown that substantial benefits can be gained by using soms instead of foms [27, 28]. Empirical results [9] show that sampling second-order mutants based on various strategies produces tests with a 10% loss on the fault revealing ability while reducing the number of the produced equivalent mutants by 80–90%. In the study presented in this paper, second-order mutants were employed in order to provide information about the first-order ones they are composed of. Thus, possible first-order equivalent mutants can be identified by observing the behaviour of the second-order ones.

2.3. The mutants' impact

The approaches studied in this paper were founded on an assertion, termed the mutants' impact, regarding killable mutants. The intuition behind the mutants' impact is that mutants able to change certain aspects of the program execution are likely to be killable. In other words, given a test, if the executions of the original and a mutant program differ, then the mutant is possibly killable. These differences are termed as the mutants' impact. The question that is raised here is how to identify these differences. To answer this, dynamic program invariants, execution program traces and methods' return values have been proposed as possible ways of identifying program execution differences [19–22].

Mutants' impact represents a difference in behaviour between the original and the mutant programs. Such differences appear during program executions and in particular after executing the mutated location up to the exit of the program [21]. Along these lines, impact on coverage measures the differences of the coverage in both the original and the mutated program versions [19, 20]. Impact on return values, measures the differences in the values returned by the public methods encountered during test execution [19, 20]. Impact on dynamic program invariants measures the number of invariants that were violated by the introduction of mutants [22].

Generally, it has been empirically found that mutants with impact are more likely to be killable than those with no impact, regardless of the impact measure [19, 20]. However, different impact measures or their combinations generally result in different mutant classifiers with variations in their effectiveness. Constructing a more effective classifier forms the objective of the present paper, which proposes the use of higher-order mutants as impact measures. Based on this novel measure, different mutants with the aforementioned approaches can be classified appropriately.

DOI: 10.1002/stvr

Table I. Mutant operators utilised by Javalanche.

Mutant operators	Description
replace numerical constant	Replaces a numerical constant instance by + 1, -1 or 0.
negate jump condition	Inserts the negation operator to logical conditional instances.
replace arithmetic operator	Replaces an arithmetic operator instance by another one.
omit method calls	Omits a method call and sets in its position a default value (the default value replaces the returned one).

2.4. Javalanche

The present study constitutes an extension of the coverage impact approach [19, 20]. In order to provide fairly comparable results, the same subjects, mutant operators and tools as in the study on the equivalent mutant classification [19, 20] were used. Thus, the present study utilises the Javalanche tool (version 0.3.6) [11], which implements the coverage impact classification method.

Javalanche [11] is a publicly available mutation testing tool for Java programs. It enables the efficient application of the mutation process by implementing a large number of optimizations in order to be scalable and efficient to real-world programs. Additionally, Javalanche has also been employed in many recent studies [17, 19–22]. Table I records details about the utilised mutant operators supported by the tool and the present study.

3. MUTANT CLASSIFICATION

This paper suggests the use of a mutant classification approach in order to isolate equivalent mutants. The proposed classification scheme utilises second-order mutants and the mutants' impact in order to highlight the majority of the killable mutants. This section addresses these techniques and concepts.

3.1. Mutation analysis using mutant classifiers

Applying first-order mutation testing entails the generation of a mutant set, referred to as the candidate mutant set. Next, this set is executed with the employed tests in order to determine its adequacy. Those mutants that are killed are removed from the candidate set of mutants. The remaining (live) mutants must then be analysed in order to produce new tests able to kill them. This process (test case production and test case evaluation) iteratively continues until all killable mutants have been killed. In view of this, the ratio of the equivalent mutants to the candidate mutant set increases, as more tests are added to the considered test suite. This is attributed to the fact that the number of equivalent mutants remains constant, whereas the number of killable ones decreases (because they are killed). Thus, if the live mutants could be automatically classified as killable and equivalent, one could claim substantial benefits by analysing only the killable ones [19, 20, 22, 29]. Such being the case, two benefits arise. First, an accurate adequacy evaluation is employed. Second, the test generation process can be effectively guided by only those killable mutants, hence saving considerable resources during the process of trying to kill those non killable mutants.

Based on the aforementioned arguments, automated mutant classification approaches have been proposed in the literature. A typical mutant classification process, steps (a)–(c) of Figure 1, involves the categorization of the set of live mutants into two disjoint sets; the possibly killable mutants and the possibly equivalent ones. Before applying the classification scheme, the set of live ones must be found, which means that the first-order mutants of a program under test must be generated (Figure 1 (a)) and executed with the available tests (Figure 1 (b)). At this point, a set of killed and a set of live mutants is created. Killed mutants do not add any value to the testing process and thus, they are discarded. The set of live mutants forms a guide towards producing new tests and thus improving the quality of the testing process. However, this set is likely composed of both equivalent and killable mutants. Therefore, the live mutants set is provided as input to the classification system. The live

DOI: 10.1002/stvr

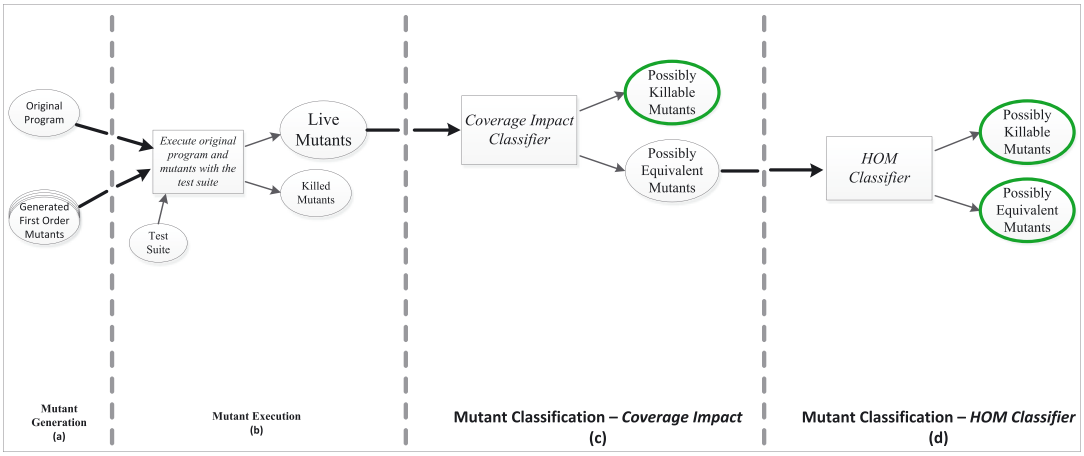

Figure 1. Mutant classification process. The live mutants are classified as possibly killable and possibly equivalent. The Isolating Equivalent Mutants (I-EQM) process works in two phases. First, the live mutants will be classified via the *coverage impact classifier* and subsequently the produced possibly equivalent mutant set will be classified by the high-order mutants classifier. The highlighted first-order mutant sets are the outcome of the I-EQM classification scheme.

mutants are then categorised as possibly killable or possibly equivalent (Figure 1 (c)). The testing process is then continued based on the possibly killable mutant set.

3.2. Mutant classification effectiveness measures

Mutant classification categorises mutants as possibly killable or possibly equivalent. Being a heuristic method, mutant classification may correctly classify some mutants and fail on others. For example, a killable mutant may be correctly classified as possibly killable, whereas an equivalent one may be incorrectly classified as possibly killable.

To distinguish between the correctly and incorrectly classified mutants, the following definitions are given, which are in accordance to the respective ones used in the information retrieval literature [30]:

- *True killable* refers to the killable mutants that are *correctly* classified as possibly killable
- *False killable* refers to the equivalent mutants that are *incorrectly* classified as possibly killable
- *True equivalent* refers to the equivalent mutants that are *correctly* classified as possibly equivalent
- *False equivalent* refers to the killable mutants that are *incorrectly* classified as possibly equivalent

To quantify the classification ability and to provide a comparison basis for the examined approaches, the following measures were utilised. These metrics are usually employed in order to compare classifiers in information retrieval experiments [30].

$$Precision = \frac{True\ killable}{True\ killable + False\ killable}$$

$$Recall = \frac{True\ killable}{True\ killable + False\ equivalent}$$

$$Accuracy = \frac{True\ killable + True\ equivalent}{True\ killable + False\ killable + True\ equivalent + False\ equivalent}$$

DOI: 10.1002/stvr

$$\boldsymbol{F_\beta} = (1+\beta^2) \times \frac{Precision \times Recall}{\beta^2 \times Precision + Recall}$$

The first two measures quantify the ability of the classifier to correctly classify killable mutants. Specifically, the precision metric quantifies the ability of the classifier to categorise correctly killable mutants, whereas the recall value measures the capability of the classifier in retrieving killable mutants. Thus, a high precision value indicates that the classifier can sufficiently distinguish between killable and equivalent mutants, whereas a high recall value shows that the classification scheme is able to recover the majority of the live killable mutants. The last two measures are used in order to better compare the examined classifiers. The accuracy metric depicts the percentage of the correctly classified mutations. The F_β score[§] constitutes a general metric that combines the precision and recall values into a single measure of overall performance and enables different weighting between the two measures. Thus, by changing the value of β, different performance scenarios could be investigated. For example, a scenario where the classification precision and recall are equally balanced corresponds to a β value of 1. In this study, three such scenarios are explored, which are described in Section 5.

Mutation testing requires the employed tests to be capable of killing all killable mutants. Mutant classification changes this requirement to killing all possibly killable mutants. Thus, mutant classification can be seen as an approximation method to mutation. The effectiveness of this approximation can be measured as the number of killable mutants that are correctly classified as such, that is, by the recall metric, because the testing process will be based only on these mutants. The efficiency of the method can be measured by the number of equivalent mutants that are incorrectly classified as possibly killable, because these mutants will be manually analysed by the tester. Thus, the methods' efficiency can be expressed by the precision[¶] value of the classifier.

In general, high precision is difficult to be achieved, but is extremely desirable for reasons of efficiency. However, if high precision is not accompanied by a relatively high recall, the process might lose some rather valuable mutants. Because low recall indicates that many killable mutants are ignored, such processes will experience losses of their strength. On the contrary, high recall solely is easily achieved by classifying most or all undetected mutants as possibly killable. Therefore, in such a case, higher testing quality is achieved at considerable cost. Thus, a combination of high precision and high recall is more suitable for a mutant classification scheme. In this manner, most of the killable mutants would be correctly classified as such and at the same time testing based on the retrieved killable mutants is deemed adequate. In a different situation, the classifier would be deficient as it would categorise many equivalent mutants as possibly killable ones (a case of a classifier with low precision) or it would classify many killable mutants as possibly equivalent ones (a classifier with low recall). Consequently, it is the reconciliation of a classifier's precision and recall scores that stipulates its success.

3.3. Mutant classification using code coverage

Classifying mutants using code coverage has been empirically found to be superior to previously proposed classifiers, such as those using dynamic invariants and return values [19, 20]. Following the suggestions of Schuler and Zeller [20], a coverage measure is determined by counting the number of times each program statement is executed during a test run. The comparison of the coverage measures of both the original's and the mutated program's execution results in the impact measure (coverage difference) of the examined mutant [19, 20].

Mutants' impact is defined based on the coverage impact as 'the number of methods that have at least one statement that is executed at a different frequency in the mutated run than in the normal run, while leaving out the method that contains the mutation' [19]. This approach is also adopted here and referred to as the *coverage impact classifier.*

[§]For non-negative real values of β.

[¶]The percentage of the equivalent mutants that are incorrectly classified as possibly killable is actually (1 – precision). Thus, higher precision indicates fewer equivalent mutants to be considered, and hence, a higher efficiency.

DOI: 10.1002/stvr

3.4. First-order mutant classification via second-order mutation

The primary purpose of this paper is the introduction of a new mutant classification scheme, hereafter referred to as *Higher Order Mutation (HOM) classifier*, which would further attenuate the effects of the equivalent mutant problem. The salient feature of the suggested approach is the employment of higher-order mutation in the classification process.

The HOM classifier categorises mutants based on the impact they have on each other. In view of this, it produces pairs of mutants by combining each examined (first-order) mutant with others. The classifier works based on the intuition that because equivalent mutants have a small effect on the state of the program, they should not have an apparent impact on the state of another mutant. Hence, a possible equivalent mutant will have a minor impact on the execution and no observable impact on the output of another mutant program. This leads to the HOM classifier hypothesis.

3.4.1. Classifier hypothesis. Let fom be a first-order mutant, umut an unclassified mutant, umut.fom the second-order mutant created by the combination of the two corresponding mutants and Killable the set of killable mutants of the considered program under test. The HOM classifier hypothesis states that if the results of the executions of the first-order mutant fom and the second-order mutant umut.fom differ, then the first-order mutant umut is killable. More formally,

$$outputOf\ (fom,\ test)\ \neq\ outputOf\ (umut.fom,\ test)\ \Rightarrow\ umut \in Killable$$

The aforementioned hypothesis forms the basis of the proposed classification scheme. Thus, if the condition of the aforementioned formula holds, then *umut* is classified as possibly killable. Otherwise, *umut* is classified as possibly equivalent. This practice is presented in Figure 2. Although this condition may not always hold, the present study suggests that it can provide substantial guidance on identifying killable and equivalent mutants.

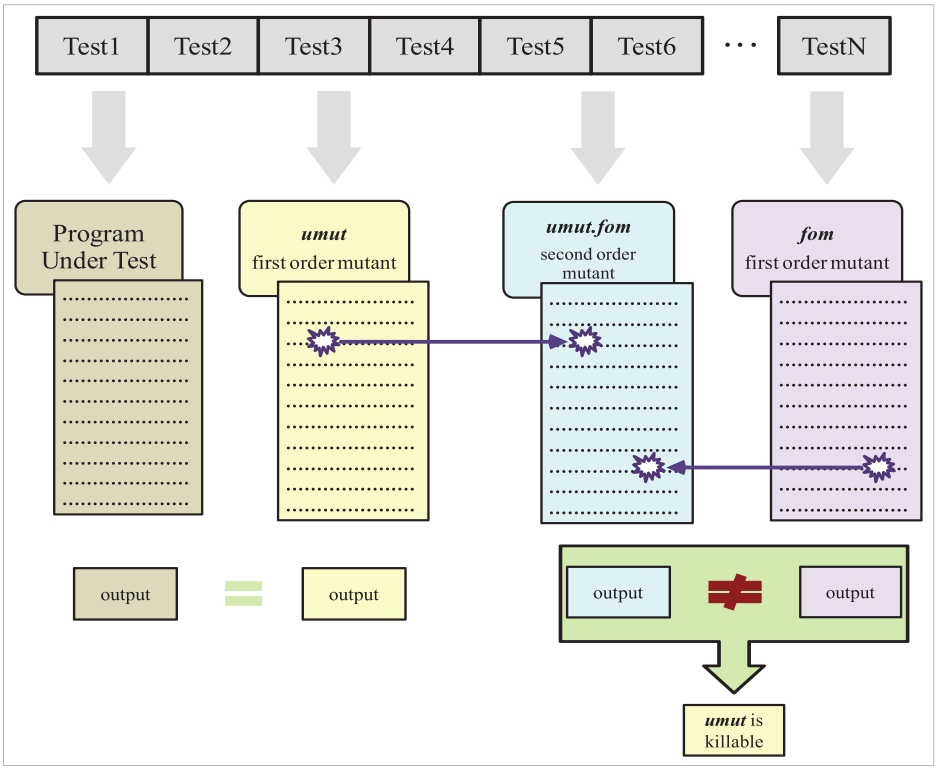

Figure 2. The unclassified first-order mutant umut is classified as possibly killable based on its impact on other fom mutants.

DOI: 10.1002/stvr

3.4.2. Mutant classification process. The mutant classification process of the *HOM Classifier* requires three main inputs. These are requisites for the evaluation of the *HOM Classifier* hypothesis condition.

- The first input is the set of the live-unclassified mutants that need to be classified. The mutants of this set will be categorised as possibly killable or possibly equivalent based on the truth or the falsehood of the classification's predicate.
- The second required input is the set of mutants, referred to as the classification mutant (CM) set that will be used for constructing the sought mutant pairs. The mutant pairs are constructed by combining these mutants with the unclassified ones.
- The final input of the classification process is the test suite, with which the second-order mutants and their corresponding first-order ones will be executed.

Finally, the algorithm pertaining to the aforementioned classification process is presented in Figure 3. The algorithm takes as inputs (a) the set of the live-unclassified first-order mutants; (b) the set of the first-order mutants of the CM set that will be used for the generation of the second-order ones; and (c) the available test suite. The mutants' impact is defined as the number of first-order ones in the CM set whose output is changed after being combined with the live-unclassified mutant.

3.4.3. Classification mutant set. Generally, the proposed classification technique determines the impact of mutants using other mutations referred to as the CM set. For example, in Figure 2, the CM set is the set of the fom mutants to be employed in order to perform an effective classification process. The question that is raised here is which mutants are suitable for supporting the mutant classification. Specifically, it has been found that not all mutations are of the same value in assessing the mutants' impact. Perhaps, by utilizing all the available mutants, one could obtain the best results. However, such an approach is prohibitive, because it requires executing all mutants with all tests for each mutant to be classified.

Considering the aforementioned issue, a necessary restriction was imposed on the utilised CM mutant set; only the mutations that appear in the same class as the aimed unclassified mutant were considered. Thus, for each mutant to be classified, a different CM set was used. This choice was based on the intuition that mutants that belong to the same class are more likely to interact and be exercised by the same test cases. It is noted that because there is no test able to kill umut, if there is no interaction between the mutants composing the second-order one (umut and fom), then the outputs of fom and umut.fom will be the same for all the employed tests.

```
1.  for each first order mutant umut in the live-unclassified set
2.    for each first order mutant fom in the CM set
3.      for each test case test in the test suite set
4.        if the result of the execution of fom with test does not exist
5.          execute fom with test and record the fom output
6.        end if
7.        generate the second order mutant umut.fom
8.        execute umut.fom with test and record the umut.fom output
9.        compare the ouputs of fom and umut.fom
10.       if the ouputs differ
11.         add umut to the set of "Possibly Killable Mutants"
12.         continue to the next mutant of the live-unclassified set
13.       end if
14.     end for each
15.   end for each
16.   add umut to the set of "Possibly Equivalent Mutants"
17. end for each
18. return the set of "Possibly Killable Mutants"
19. return the set of "Possibly Equivalent Mutants"
```

Figure 3. HOM classification process.

DOI: 10.1002/stvr

3.4.4. HOM classifier variations. The work presented here considers three variations of the HOM classifier, namely, HOM classifier (all foms), HOM classifier (killed foms) and HOM classifier (same method foms). These approaches differ in the sets of mutants that will be used for the creation of the second-order ones. Recall that the employed CM set is the one composed of the mutants that belong to the same class as the aimed unclassified mutant. The HOM classifier (all foms) considers the whole CM set, whereas the other approaches utilise only a specific subset. The HOM classifier (killed foms) uses only those mutants that have been killed by the employed tests, whereas the HOM classifier (same method foms) employs only those that appear in the same method as the aimed unclassified mutant. The reason of using only the already killed mutants is that because these mutants are more sensitive to the utilised tests (easy-to-kill) than the live ones, they will be more sensitive to the impact of the examined ones, too. Similarly, using mutants belonging to the same method as the examined mutant increases the chances of these mutants to be coupled. Besides the reasons mentioned earlier, their application was empirically found to be sound. Furthermore, constructing the sought second-order mutants based on the killed mutant set or the mutants of the same method results in reducing the overall computational cost of the technique because of the reduced number of mutant combinations and their respective executions.

3.5. I-EQM mutant classification

In addition to the previously described techniques, the possibility of employing a combined strategy is also investigated. To this end, the I-EQM classification scheme is proposed. The I-EQM classifier constitutes a combination of the coverage impact classifier [19, 20] and the variations of the HOM classifier. The utilization of the coverage impact classifier is based on its evaluation, which presented the best results among those examined in previous studies [19, 20]. Specifically, it resulted in a classification precision of 75% and a recall value of 56%. In addition, the coverage impact classifier manages to successfully classify most of the mutants that are correctly classified by the rest of the examined approaches [19, 20]. Because the corresponding precision measure of the coverage impact classifier can be considered as an accurate one, that is, it does not misclassify many equivalent mutants; effort should be put into improving the recall measure. Achieving higher recall results in considering a larger number of killable mutants, thus strengthening the testing process.

In order to effectively combine different classifiers, they must adhere to the following two requirements: first, the precision of both classification schemes must be reasonably high. Second, they should classify different non-equivalent mutants as possibly killable. In other words, both produced sets must be accurate and their intersection must be of small size. As a result, the precision of the combined classifiers will not be greatly affected, whereas their recall will be significantly improved. Combining classifiers has been attempted by Schuler and Zeller [19, 20] but without success.

The I-EQM classification process is summarised in Figure 1. After obtaining the set of live mutants (Figure 1 (a), (b)), the coverage impact classifier is employed in order to perform the first step of the I-EQM's classification procedure (Figure 1 (c)). This step will produce the set of the possibly killable and the possibly equivalent mutants of the coverage impact classifier. Next, the HOM classifier is applied to the possibly equivalent mutant set that was previously generated (Figure 1 (d)). This phase will create the set of the possibly killable mutants and the set of the possibly equivalent ones of the HOM classifier. The I-EQM classifier's resulting set of possibly killable mutants is the union of the possibly killable mutant set of the coverage impact classifier and the corresponding killable set of the HOM classifier. The possibly equivalent mutant set of the I-EQM classifier is the one generated by the HOM classifier. The aforementioned sets appear highlighted in Figure 1. It should be noted that the order in which the coverage impact and HOM classifiers are applied to the set of live mutants is indifferent, that is, the resulting sets of the possibly killable and the possibly equivalent mutants would be the same regardless of the application order of the two classifiers. This is due to the fact that the resulting set of the possibly killable mutants will be composed of the mutants that are categorised as such by at least one of the combined classifiers. The possibly equivalent mutant set will contain the mutants that are categorised as such by all the combined classifiers. This can be also generalised to a classification process that includes more than two classifiers.

DOI: 10.1002/stvr

Table II. Subject program details.

Subject Programs	Description	Version	Lines of code	Test cases
ASPECTJ[a]	AOP extension to Java	cvs: 2007-09-15	25,913	336
BARBECUE	Bar code creator	svn: 2007-11-26	4,837	153
COMMONS	Helper utilities	svn: 2009-08-24	19,583	1608
JAXEN	XPath engine	svn: 2008-12-03	12,438	689
JODA-TIME	Date and time library	svn: 2009-08-17	25,909	3497
JTOPAS	Parser tools	1.0 (SIR)	2,031	128
XSTREAM	XML object serialization	svn: 2009-09-02	16,791	1122

[a]Only the *org.aspectj.ajdt.core* package was considered.

Table III. Mutant samples profile with regard to subject program.

	Considered mutants		Manually classified mutants					
			Killable			Equivalent		
Subject Programs	Number of mutants	Reached mutants	Set1	Set2	Set1+Set2	Set1	Set2	Set1+Set2
ASPECTJ	6613	2878	15 (75%)	6 (60%)	21 (70%)	5 (25%)	4 (40%)	9 (30%)
BARBECUE	3283	1603	14 (70%)	9 (90%)	23 (77%)	6 (30%)	1 (10%)	7 (23%)
COMMONS	8693	8305	6 (30%)	6 (60%)	12 (40%)	14 (70%)	4 (40%)	18 (60%)
JAXEN	6619	3944	10 (50%)	7 (70%)	17 (57%)	10 (50%)	3 (30%)	13 (43%)
JODA-TIME	5322	4197	14 (70%)	7 (70%)	21 (70%)	6 (30%)	3 (30%)	9 (30%)
JTOPAS	1676	1402	10 (50%)	4 (40%)	14 (47%)	10 (50%)	6 (60%)	16 (53%)
XSTREAM	2156	1730	8 (40%)	5 (50%)	13 (43%)	12 (60%)	5 (50%)	17 (57%)
TOTAL	**34362**	**24059**	**77 (55%)**	**44 (63%)**	**121 (58%)**	**63 (45%)**	**26 (37%)**	**89 (42%)**

4. SUBJECT PROGRAMS AND BENCHMARK MUTANT SETS

The evaluation of the examined classification schemes is based on a set of seven open-source projects. The motivation behind their selection is twofold. First, some of these projects have been utilised in various mutation testing studies [17, 19, 20, 22] and thus, can be considered as benchmarks. Moreover, the previous mutant classification studies [19, 20] consider the same program set, hence, a comparison between their approach and the ones proposed in the present paper can be directly performed. Secondly, these programs constitute real-world examples and therefore would provide valuable insights about the practical applicability and the efficacy of the examined mutant classification techniques. Details about the subject program names, their respective description, program versions, lines of code and accompanied test cases are recorded in Table II.

In order to investigate the effectiveness of the proposed approaches, the mutant set used by the previous mutant classification approaches [19, 20] was employed, hereafter referred to as 'control mutant set 1'. The control mutant set 1 is comprised of 140 manually classified mutants, each one belonging to a different class of the subject programs, with a total number of 20 mutants per test subject. To avoid overfitting or coincidental results, a second set of mutants was also considered. The second set, referred to as the 'control mutant set 2', was independently constructed by the authors of the present paper in a similar fashion to the previous studies [19, 20]. Specifically, for each test subject, 10 live mutants were randomly chosen for manual classification, each one belonging to a different class. Thus, the control mutant set 2 is comprised of 70 manually classified mutants, with a total number of 10 mutants per test subject.

Details about the considered mutant samples with respect to the examined programs and utilised mutant operators are recorded in Tables III and IV. The 'Considered Mutants' column of Table III refers to the number of considered mutants, among those generated by the Javalanche framework. It must be mentioned that the considered mutants are those employed by the present experiment and belong to the same classes as the manually classified ones. The first sub-column presents the total number of the corresponding mutants, whereas the second one the number of mutants that are

DOI: 10.1002/stvr

Table IV. Mutant samples profile with regard to mutant operator.

Mutant Operators	Number of mutants			Killable mutants			Equivalent mutants		
	Set1	Set2	Set1+Set2	Set1	Set2	Set1+Set2	Set1	Set2	Set1+Set2
replace numerical constant	78 (56%)	48 (69%)	126 (60%)	34 (44%)	27 (56%)	61 (48%)	44 (56%)	21 (44%)	65 (52%)
negate jump condition	12 (9%)	8 (11%)	20 (10%)	10 (83%)	7 (87.5%)	17 (85%)	2 (17%)	1 (12.5%)	3 (15%)
replace arithmetic operator	7 (5%)	2 (3%)	9 (4%)	3 (43%)	1 (50%)	4 (44%)	4 (57%)	1 (50%)	5 (56%)
omit method calls	43 (31%)	12 (17%)	55 (26%)	30 (70%)	9 (75%)	39 (71%)	13 (30%)	3 (25%)	16 (29%)

DOI: 10.1002/stvr

executed by the available test cases. Finally, the 'Manually Classified Mutants' column presents the number of the equivalent and killable mutants among the manually classified ones. The columns of Table IV record the number of mutants, the number of killable and equivalent ones per set and mutant operator, respectively. From Table IV, it can be observed that the 'replace numerical constant' and 'replace arithmetic operator' operators produce more than 20% equivalent mutants than the 'negate jump condition' and 'omit method calls' operators.

Interestingly, the two samples have different distributions of killable and equivalent mutants. The first one is composed of 55% of killable mutants, whereas the second one of 63%. Consequently, the second sample contains more killable mutants (as a ratio) than the first sample when considering them with respect to each program or operator. The difference between the two sets can be attributed to various factors, such as the random selection, the sample sizes or the researchers performing the classification. Because both sets were selected from the same programs, the population distribution can be estimated based on both samples (Set1 + Set2). Therefore, the examined population is estimated to be consisting of 58% of killable mutants and 42% of equivalent mutants. Recall that the examined population is the mutants that have been left alive after their execution with the employed tests.

5. EVALUATION

The present study investigates a new aspect of higher-order mutants, their ability to classify first-order ones. This attribute constitutes the basis of the HOM and the I-EQM classifiers, described in Section 3. The present section describes the empirical evaluation of these approaches and directly compares them to the current state of the art, coverage impact classifier. [19, 20]. Additionally, a revalidation of the coverage impact classifier is also presented.

5.1. Research objectives

The following research points summarise the primary purpose of the presented empirical evaluation:

- Can second-order mutation provide adequate guidance in equivalent mutant isolation, that is, how effective are the HOM and I-EQM classifiers in terms of their recall and precision metrics?
- Do the HOM and I-EQM classifiers perform better than the coverage impact classifier?
- How stable is the categorization ability of the HOM, I-EQM and coverage impact classifiers across the two different mutant sets? Are there any important differences between the two sets?

5.2. Experimental setup

In order to deal with the aforementioned research points, the recall and precision values of the examined techniques were determined. To this end, the HOM, I-EQM and coverage impact classifiers are applied to both considered control mutant sets with the aim of classifying them as possibly killable or possibly equivalent ones.

The conducted experiment[‖] uses the Javalanche mutation testing framework in order to produce and execute the sought mutants. Javalanche was employed to generate the first-order mutants of the classes that the mutants of the control mutant sets belong to. These mutants were then executed with the available test cases[**] and their coverage impact was determined. For the application of the *coverage impact* technique, the tool's default execution settings were used. For the rest of the examined approaches, mutant execution options were set as follows: (a) 100 s for timeout limit[††] and (b) execution of all tests with all mutants. Because Javalanche does not support second-order mutation, the applied process for generating second-order mutants is presented in Figure 4. Initially, the source

[‖]The experiment was conducted on a single machine (CPU: i3 – 2.53GHz (2 processor cores), RAM: 3GB), running Windows 7 x64 and Oracle Java 6 with default jvm configuration (Javalanche by default sets the -Xmx option to 2048 megabytes).

[**]The same tests as in previous studies [19, 20] were utilised.

[††]If mutant execution time exceeds this limit, the mutant is treated as killed, provided that the original program has successfully terminated.

DOI: 10.1002/stvr

```
1.  for each first order mutant umut of the control mutant sets
2.    generate the first order mutants of umut that belong to the same class
3.    for each generated second order mutant umut.fom
4.      if its combined mutants affect the same source code position and are
          produced by the same mutant operator
5.        remove umut.fom from the resulting set of second order mutants
6.      end if
7.    end for each
8.    return the resulting set of second order mutants
9.  end for each
```

Figure 4. Generation process of second-order mutants.

code of each first-order mutant of the control mutant sets was created manually. This resulted in a total of 210 different class versions. Next, Javalanche was employed to produce mutants for each of these 210 classes. Note that this process yields second-order mutants. Mutants belonging to the same position[‡‡] as the examined one and produced by the same mutant operator were discarded from the considered mutant set. Should such an action not be applied, the impact of the examined mutant would be impossible to be assessed. Based on this process, all the required mutant pairs, that is, second-order mutants, were generated. Each pair is composed of the examined mutant and another one belonging to the same class. Finally, the *HOM* classification process, as presented in Figure 3, was performed.

In summary, for each mutant of the control mutant sets, the following procedure was employed:

- The first-order mutants of the appropriate class were generated and executed via the available test cases.
- The appropriate second-order mutants were generated and executed via the available test cases.
- The outputs of the first-order mutants and the second-order ones were compared in order to classify the examined mutants.

The aforementioned process was performed for the HOM and I-EQM classifiers for all their respective variants, that is, all foms, killed foms and same method foms. Recall that the basic difference of these approaches is the set of second-order mutants they rely on. In the case of the I-EQM classifier, presented in Figure 1, the coverage impact classifier classifies the examined mutants as possibly killable and possibly equivalent ones; those classified as possibly equivalent are subsequently categorised based on the HOM classifier.

A comparison between mutant classifiers was attempted based on the *accuracy* and the F_β measure scores, metrics usually used in comparing classifiers in information retrieval experiments [30]. These measures were utilised to validate in a more typical manner the differences between the classifiers. In order to avoid the influence of outliers, the median values were used. Regarding the F_β measure, three possible scenarios are examined. Here, it should be noted that high recall values indicate that more killable mutants are to be considered, hence leading to a more thorough testing process. High precision indicates that fewer equivalent mutants are to be examined, leading to an efficient process. The first scenario refers to the case where a balanced importance between the recall and precision metrics is desirable. This case is realised by evaluating the F_β measure with $\beta = 1$. The second scenario emphasises on the recall value and is achieved by assigning a value of $\beta = 2$. The last scenario, which is accomplished by using $\beta = 0.5$, weights precision higher than recall.

In the present experiment, special care was taken to handle certain cases because of some inconsistencies of the utilised tool. Specifically, it was observed that in the cases of execution timeouts, the tool gave different results when some mutants were executed in isolation than together with others. To circumvent this problem, in the cases of the HOM and I-EQM classifiers, when mutants were categorised as possibly killable because of timeout conditions (if either the first-order mutant or the second-order one results in a timeout), the corresponding mutants of the considered mutant

[‡‡]Position refers to the exact part of the original source code that differs from the mutant programs.

DOI: 10.1002/stvr

pairs were isolated and re-executed individually with an increased timeout limit. Similarly, in the case of the coverage impact, the mutants that were classified differently with respect to the previous study [19], were re-executed in isolation with a greater timeout limit. Although the previous study's results [19] could be used, this would constitute a potential threat to the validity of the comparison because of the dissimilar employed environments and the use of the control mutant set 2. Other special considerations include issues concerning the comparison of the programs' output. Note that such a comparison is performed between every first-order mutant and its respective second-order one. Many programs had outputs dependent on each specific execution. Execution outputs containing time information, e.g., the Joda-Time and AspectJ test subjects, or folder locations, for e.g., the JTopas program, are examples of such cases. To effectively handle these situations, the execution dependent portions of the considered outputs were replaced with predefined values via the employment of appropriate regular expressions.

5.3. Results

The evaluation results of the coverage impact classifier are recorded in Table V. The table presents the classification precision and recall for the examined control mutant sets and test subjects. The columns named 'Killable Set1', 'Killable Set2', 'Equivalent Set1' and 'Equivalent Set2' correspond to the sets of killable and equivalent mutants of the considered control mutant sets. Their subcolumns present details about the number of the correctly and incorrectly classified mutants. Finally, the precision and recall values of the classifier are recorded in the last two columns of the table. On average, the coverage impact method achieves a precision of 72% and a recall of 66% for the control mutant set 1. It is noted that these results were calculated by executing Javalanche with the settings described in the previous subsection. Hence, the difference between the aforementioned results and the ones reported in the respective studies of Schuler and Zeller [19, 20] is attributed to the execution environment used for performing the present study. Regarding the second control mutant set, the obtained precision score is 76% and the corresponding recall value is 64%.

The respective results of the HOM classifier are recorded in Table VI and Table VII, for the corresponding control mutant sets. Both tables have similar structure to Table V, except that each figure of each cell refers to a different variation of the method. Thus, the first figure corresponds to the HOM classifier (all foms) variation, the second one to the HOM classifier (killed foms) and the third one to the HOM classifier (same method foms). The last two columns of each table present the precision and recall metrics per subject. It can be seen that, for the control mutant set 1, the HOM classifier (all foms) achieves a precision value of 69% and a recall value of 57%, the HOM classifier (killed foms) variation realises a precision of 70% and a recall of 55% and the HOM classifier (same method foms)c variation a precision of 71% and a recall of 45%, respectively. Considering the control mutant set 2, the precision of the HOM classifier (all foms) technique is 88% and the recall is 50%, the corresponding values of the HOM classifier (killed foms) variation are 88% and 48%, and the ones obtained for the HOM classifier (same method foms) are 84% and 36%, respectively.

The I-EQM classifier's experimental evaluation is depicted in Table VIII and Table IX, which present details about the precision and the recall metrics of the I-EQM's variations for the examined control mutant sets. Note that the same structure as Table VI and Table VII is used. For the control mutant set 1, the average precision of the I-EQM's variation that employs the HOM classifier (all foms) method is 67% and its recall is 83%, the corresponding values of using the HOM classifier (killed foms) variation are 68% and 83% and the ones obtained by utilizing the HOM classifier (same method foms) are 69% and 81%, respectively.[§§] Finally, with respect to the second control mutant set, the I-EQM classifier's variation that utilises the HOM classifier (all foms) technique realises a precision score of 76% and a recall value of 80%, the one that employs the HOM classifier (killed foms) variation achieves a precision of 76% and a recall of 77%, whereas the variation that uses the HOM classifier (same method foms) achieves a precision and recall value of 75%.

[§§]The difference between these results and the previously published ones [29] is due to the coverage impact method's re-evaluation; the previous results were based on the reported results of Schuler and Zeller. [19], whereas the new ones on the method's re-evaluation (Section 5.*B*.).

DOI: 10.1002/stvr

Table V. Mutant classification using the coverage impact classifier.

	Killable Set1		Equivalent Set1		Killable Set2		Equivalent Set2		Possibly killable Set1		Possibly killable Set2	
Subject Programs	True Killable	False Equiv.	True Equiv.	False Killable	True Killable	False Equiv.	True Equiv.	False Killable	Precision %	Recall %	Precision %	Recall %
ASPECTJ	13	2	0	5	6	0	0	4	72%	87%	60%	100%
BARBECUE	10	4	3	3	8	1	1	0	77%	71%	100%	89%
COMMONS	0	6	12	2	1	5	4	0	0%	0%	100%	17%
JAXEN	4	6	7	3	3	4	3	0	57%	40%	100%	43%
JODA-TIME	11	3	3	3	4	3	0	3	79%	79%	57%	57%
JTOPAS	8	2	10	0	2	2	6	0	100%	80%	100%	50%
XSTREAM	5	3	8	4	4	1	3	2	56%	63%	67%	80%
TOTAL	**51**	**26**	**43**	**20**	**28**	**16**	**17**	**9**	**72%**	**66%**	**76%**	**64%**

DOI: 10.1002/stvr

Table VI. Mutant classification using the HOM classifier – control mutant set 1 (w.r.t. all foms – killed foms – same method foms).

	Killable Set1						Equivalent Set1						Possibly killable Set1					
Subject programs	True Killable			False Equivalent			True Equivalent			False Killable			Precision %			Recall %		
ASPECTJ	8	8	4	7	7	11	2	2	4	3	3	1	73	73	80	53	53	27
BARBECUE	9	7	7	5	7	7	5	5	5	1	1	1	90	88	88	64	50	50
COMMONS	4	4	4	2	2	2	10	10	12	4	4	2	50	50	67	67	67	67
JAXEN	6	6	3	4	4	7	7	7	8	3	3	2	67	63	60	60	60	30
JODA-TIME	9	9	9	5	5	5	4	5	5	2	1	1	82	90	90	64	64	64
JTOPAS	3	3	3	7	7	7	10	10	10	0	0	0	100	100	100	30	30	30
XSTREAM	5	5	5	3	3	3	5	6	5	7	6	7	42	45	42	63	63	63
TOTAL	**44**	**42**	**35**	**33**	**35**	**42**	**43**	**45**	**49**	**20**	**18**	**14**	**69**	**70**	**71**	**57**	**55**	**45**

Table VII. Mutant classification using the HOM classifier – control mutant set 2 (w.r.t. all foms – killed foms –same method foms).

	Killable Set2						Equivalent Set2						Possibly killable Set2					
Subject Programs	True Killable			False Equivalent			True Equivalent			False Killable			Precision %			Recall %		
ASPECTJ	5	5	4	1	1	2	3	3	3	1	1	1	83	83	80	83	83	67
BARBECUE	2	2	2	7	7	7	1	1	1	0	0	0	100	100	100	22	22	22
COMMONS	3	3	2	3	3	4	3	3	3	1	1	1	75	75	67	50	50	33
JAXEN	4	4	3	3	3	4	3	3	3	0	0	0	100	100	100	57	57	43
JODA-TIME	4	4	3	3	3	4	3	3	3	0	0	0	100	100	100	57	57	43
JTOPAS	1	0	0	3	4	4	6	6	6	0	0	0	100	0	0	25	0	0
XSTREAM	3	3	2	2	2	3	4	4	4	1	1	1	75	75	67	60	60	40
TOTAL	**22**	**21**	**16**	**22**	**23**	**28**	**23**	**23**	**23**	**3**	**3**	**3**	**88**	**88**	**84**	**50**	**48**	**36**

Table VIII. Mutant classification using the Isolating Equivalent Mutants classifier – control mutant set 1 (w.r.t. all foms – killed foms – same method foms).

	Killable Set1						Equivalent Set1						Possibly killable Set1					
Subject Programs	True Killable			False Equivalent			True Equivalent			False Killable			Precision %			Recall %		
ASPECTJ	14	14	13	1	1	2	0	0	0	5	5	5	74	74	72	93	93	87
BARBECUE	12	12	12	2	2	2	2	2	2	4	4	4	75	75	75	86	86	86
COMMONS	4	4	4	2	2	2	8	8	10	6	6	4	40	40	50	67	67	67
JAXEN	6	6	5	4	4	5	6	6	7	4	4	3	60	60	63	60	60	50
JODA-TIME	13	13	13	1	1	1	2	3	3	4	3	3	76	81	81	93	93	93
JTOPAS	9	9	9	1	1	1	10	10	10	0	0	0	100	100	100	90	90	90
XSTREAM	6	6	6	2	2	2	3	4	3	9	8	9	40	43	40	75	75	75
TOTAL	**64**	**64**	**62**	**13**	**13**	**15**	**31**	**33**	**35**	**32**	**30**	**28**	**67**	**68**	**69**	**83**	**83**	**81**

These results provide evidence that the HOM classifier hypothesis is an appropriate mutant classification property. Therefore, the employment of second-order mutation can be beneficial in isolating equivalent mutants. Additionally, the results of the HOM classifier's variations indicate that the utilization of only the killed first-order mutants as the CM set achieves approximately the same classification effectiveness as the employment of all the generated first-order mutants. Regarding the HOM classifier (same method foms) technique, it is less effective than the other two variations, but in most cases is more efficient because of the reduced size of the considered CM set. With these facts evident, because the HOM classifier (killed foms) is more efficient than the HOM classifier (all foms), its use is advisable. Finally, compared with the coverage impact classifier, the HOM classifier attains lower precision and recall values, indicating that the coverage impact classifier is a better one.

DOI: 10.1002/stvr

Table IX. Mutant classification using the Isolating Equivalent Mutants classifier – control mutant set 2 (w.r.t. all foms – killed foms – same method foms).

	Killable Set2						Equivalent Set2						Possibly killable Set2					
Subject Programs	True Killable			False Equivalent			True Equivalent			False Killable			Precision %			Recall %		
ASPECTJ	6	6	6	0	0	0	0	0	0	4	4	4	60	60	60	100	100	100
BARBECUE	8	8	8	1	1	1	1	1	1	0	0	0	100	100	100	89	89	89
COMMONS	3	3	2	3	3	4	3	3	3	1	1	1	75	75	67	50	50	33
JAXEN	5	5	5	2	2	2	3	3	3	0	0	0	100	100	100	71	71	71
JODA-TIME	5	5	5	2	2	2	0	0	0	3	3	3	63	63	63	71	71	71
JTOPAS	3	2	2	1	2	2	6	6	6	0	0	0	100	100	100	75	50	50
XSTREAM	5	5	5	0	0	0	2	2	2	3	3	3	63	63	63	100	100	100
TOTAL	**35**	**34**	**33**	**9**	**10**	**11**	**15**	**15**	**15**	**11**	**11**	**11**	**76**	**76**	**75**	**80**	**77**	**75**

Table X. Mutant classification using the coverage impact classifier on both samples.

	Killable Set1+Set2		Equivalent Set1+Set2		Possibly killable Set1+Set2	
Subject Programs	True Killable	False Equivalent	True Equivalent	False Killable	Precision %	Recall %
ASPECTJ	19	2	0	9	68%	90%
BARBECUE	18	5	4	3	86%	78%
COMMONS	1	11	16	2	33%	8%
JAXEN	7	10	10	3	70%	41%
JODA-TIME	15	6	3	6	71%	71%
JTOPAS	10	4	16	0	100%	71%
XSTREAM	9	4	11	6	60%	69%
TOTAL	**79**	**42**	**60**	**29**	**73%**	**65%**

Table XI. Mutant classification using Isolating Equivalent Mutants classifier on both samples (w.r.t. all foms – killed foms – same method foms).

	Killable Set1+Set2						Equivalent Set1+Set2						Possibly killable Set1+Set2					
Subject Programs	True Killable			False Equiv			True Equiv			False Killable			Precision %			Recall %		
ASPECTJ	20	20	19	1	1	2	0	0	0	9	9	9	69	69	68	95	95	90
BARBECUE	20	20	20	3	3	3	3	3	3	4	4	4	83	83	83	87	87	87
COMMONS	7	7	6	5	5	6	11	11	13	7	7	5	50	50	55	58	58	50
JAXEN	11	11	10	6	6	7	9	9	10	4	4	3	73	73	77	65	65	59
JODA-TIME	18	18	18	3	3	3	2	3	3	7	6	6	72	75	75	86	86	86
JTOPAS	12	11	11	2	3	3	16	16	16	0	0	0	100	100	100	86	79	79
XSTREAM	11	11	11	2	2	2	5	6	5	12	11	12	48	50	48	85	85	85
TOTAL	**99**	**98**	**95**	**22**	**23**	**26**	**46**	**48**	**50**	**43**	**41**	**39**	**70**	**71**	**71**	**82**	**81**	**79**

Considering the I-EQM classifier's results, it is evident that it forms an effective combinatory strategy. It achieves to retrieve more than 80% of the killable mutants with a reasonably high precision of approximately 70% for each control mutant set. This high retrieval capability is attributed to the ability of the HOM classifier to classify different killable mutants than the coverage impact classifier. As a consequence, their combination enhances the corresponding recall value by nearly 20%, meaning that approximately 20% more killable mutants are to be considered. To better compare these two classifiers, Table X presents the classification precision and recall values of the coverage impact technique when applied to the union of the control mutant sets while Table XI the same results for the I-EQM's variations. Note that the tables are structured in a similar manner as the previously described ones. On average, the coverage impact method achieves a classification precision of 73% and a recall value of 65%. The I-EQM classifier's variation that uses the HOM classifier (all

DOI: 10.1002/stvr

foms) approach realises a precision score of 70% and a recall value of 82%, the one that employs the HOM classifier (killed foms) variation achieves a precision of 71% and a recall of 81%, and the variation that utilises the HOM classifier (same method foms) achieves a precision of 71% and a recall value of 79%. These results indicate that all three variations realise a high recall value, with a maximum recall of 82% while achieving a reasonably high precision score, with a minimum of 70%. Concisely, the I-EQM method achieves a superior recall value while attaining a small loss of its precision. In particular, the I-EQM technique realises a gain of 16% over the coverage impact's recall metric for a loss of 2% on its precision.

The accuracy and the F_β measure scores of the examined approaches with respect to control mutant sets 1 and 2 are given in Figures 5 and 6. The left part of the figures presents the accuracy metric and the right one the examined F_β measure scores. Note that the reported results correspond to median values. In these figures, the 'All Mutants' series refers to a naive classifier that categorises all mutants as possibly killable. In such a case, it achieves a precision of 55% and 63% for the control mutant set 1 and 2, respectively. In both sets, the recall value is equal to 100%. The I-EQM and HOM methods refer to their respective killed foms variation. It is noted that the all foms and killed foms variations were found to have similar results. From the findings of Figure 5 referring to the accuracy measure, it can be observed that the I-EQM classification technique is the most accurate one, followed by the coverage impact and the HOM classifiers, which have similar performance, and lastly, the ALL mutants approach. Similar results are obtained for the control mutant set 2, as shown in the left part of Figure 6.

As mentioned in Section 5, by evaluating the F_β measure with different values of β, three possible scenarios are examined. Note that high recall values indicate a more thorough testing process, whereas high precision a more efficient one. The first scenario, which considers recall and precision of equal importance, is represented by the F_1 measure (β equal to 1). The corresponding results, depicted in the right part of Figures 5 and 6 suggest that the I-EQM approach outperforms the rest. The second scenario, described by the F_2 measure (β equal to 2), emphasises on the recall value. It

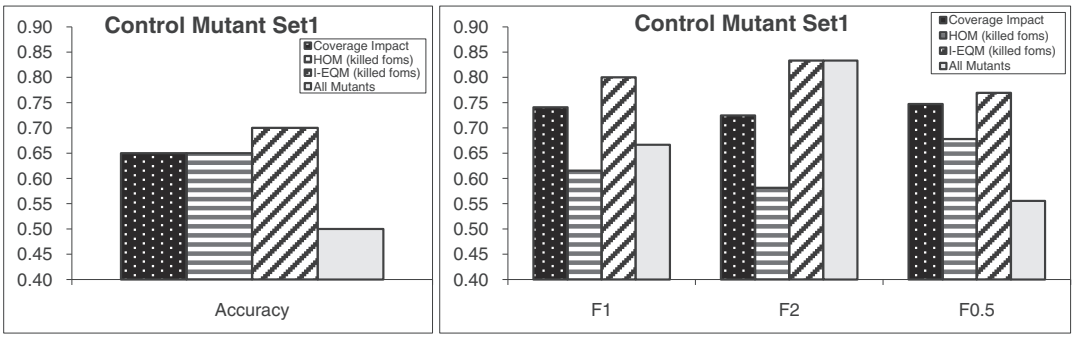

Figure 5. Mutant classifiers comparison for control mutant set 1.

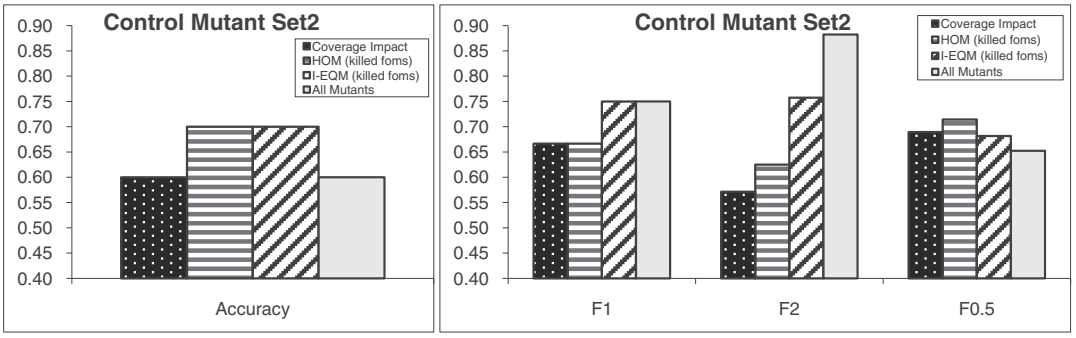

Figure 6. Mutant classifiers comparison for control mutant set 2.

DOI: 10.1002/stvr

can be seen that the I-EQM classification approach achieves by far better results than the coverage impact and *HOM* techniques for both examined control mutant sets, but worse than the ALL mutants classifier for the control mutant set 2. This is expected, because the ALL mutants categorises all examined mutants as possibly killable. Finally, the last scenario, which limits the selection of equivalent mutants, favours the precision metric over the recall one. For this case, described by the $F_{0.5}$ measure (β equal to 0.5), the I-EQM classifier achieves better results than the rest of the examined approaches for the control mutant set 1. Regarding the control mutant set 2, the I-EQM classifier achieves approximately the same (slightly worse) results with the coverage impact. However, in this case, the HOM classifier scores better than the rest.

Generally, the I-EQM classification method provides better results than the coverage impact one with respect to the accuracy and all the considered scenarios. Despite the variation on the classification ability of the HOM approach between the two samples, the I-EQM achieves a higher accuracy for both examined sets. This fact indicates that the two methods detect-classify different mutants. Hence, their combination is effective. To this end, Figure 7 presents the overall (i.e., with respect to the union of the control mutant sets) accuracy and F_β measure scores of the classification approaches. Again, the results refer to the killed foms variations of the I-EQM and HOM classifiers using median values. From the corresponding findings, it can be argued that the I-EQM constitutes a better mutant classifier than the rest of the examined ones.

5.4. Stability of the classifiers

In order to examine the stability of the considered classifiers across the test subjects, the standard deviation regarding the precision and the recall metrics with respect to the union of the control mutant sets was calculated per examined technique. Figure 8 displays these findings; the columns of the charts depict the mean precision and recall values per utilised program, for each of the examined approaches. The vertical bars represent the corresponding values that lie within one standard

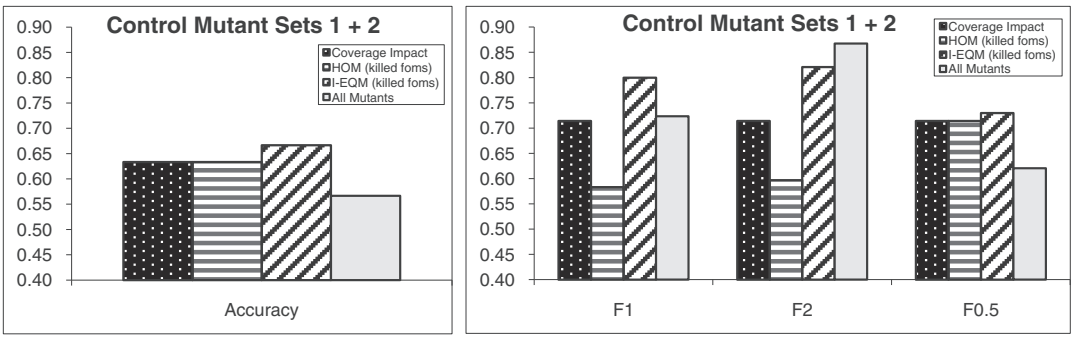

Figure 7. Mutant classifiers comparison for control mutant sets 1 and 2.

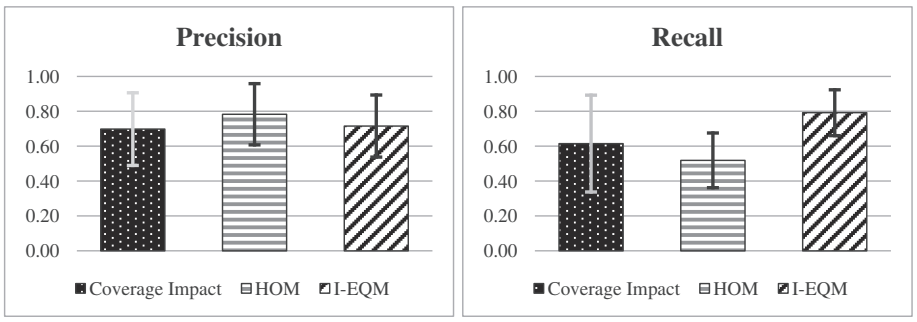

Figure 8. Mean and standard deviation values of the precision and recall metrics for both control mutants sets.

DOI: 10.1002/stvr

deviation of the mean. Again, the presented results for the HOM and I-EQM techniques refer to the killed foms variation.

With regard to precision, the coverage impact technique presents the greatest variation among the examined methods with a standard deviation of 21%. The HOM and I-EQM classifiers demonstrate similar variation levels of approximately 18%. With respect to the recall metric, the coverage impact approach experiences a variation level of 28%, whereas the HOM and I-EQM classifiers 16% and 13%, respectively. From the presented data, it can be concluded that the I-EQM approach tends to be more stable than the rest of the examined techniques. To visualise the variation among the precision and recall values of the classifiers, Figure 9 presents two groups of boxplots for the corresponding measures. It can be seen that the coverage impact and I-EQM techniques have a similar spread regarding their precision values with coverage impact having a slight advantage. However, regarding the recall values, the I-EQM is clearly better.

Comparing the three classification approaches based on the results presented both here and in the previous subsection, it becomes evident that the I-EQM technique manages to provide better recall values with only a minor loss of precision and with the highest level of stability for the examined test subjects.

5.5. Statistical analysis

In order to investigate whether the previously described differences among the examined classifiers are statistically significant, the Wilcoxon Signed Rank Test was employed per compared technique. The Wilcoxon Signed Rank Test is a non-parametric test for two paired samples that tests whether or not the two populations from which the corresponding samples are drawn, are identical. A non-parametric test was utilised, instead of a parametric one, because of the fact that it is based on no distributional assumptions for the considered data observations. A series of two-tailed hypothesis tests were performed to investigate if the HOM and I-EQM classifiers have similar effectiveness to the coverage impact technique, regarding the precision and the recall metrics. For example, the hypotheses that were tested in the case of the HOM classifier, regarding its precision metric, are the following:

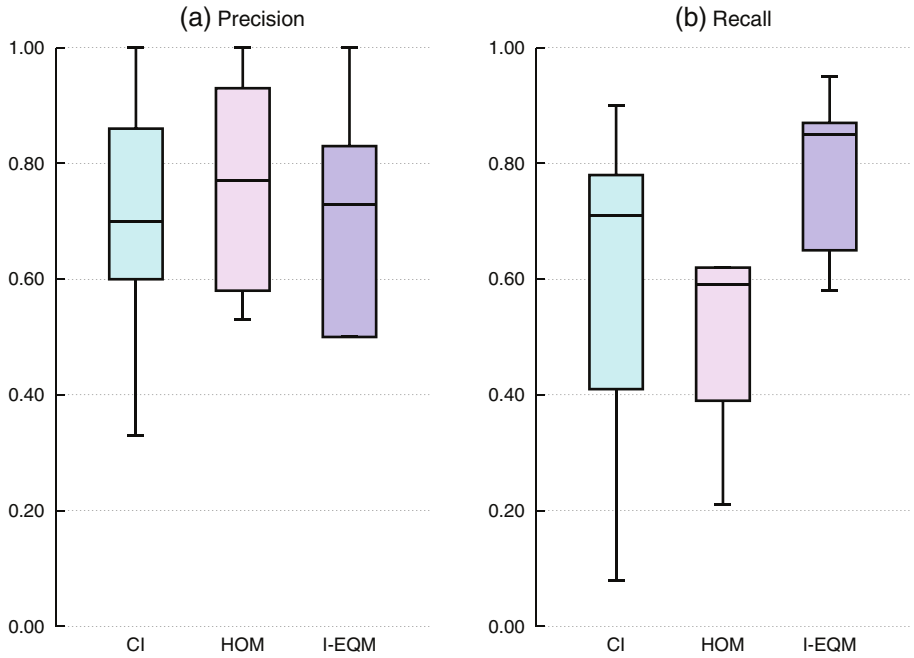

Figure 9. Variation among the precision and recall values of the examined classifiers for both control mutants sets.

DOI: 10.1002/stvr

Table XII. Statistical significance based on the Wilcoxon signed rank test.

	Compared techniques	p-value (two-tailed)
Precision	CI versus HOM	0.094
	CI versus I-EQM	0.563
	HOM versus I-EQM	**0.031**
Recall	CI versus HOM	0.469
	CI versus I-EQM	**0.016**
	HOM versus I-EQM	**0.031**

- H_0 : The *HOM* and *Coverage Impact* classifiers perform the same with regard to the precision metric.
- H_1 : The *HOM* and *Coverage Impact* classifiers perform differently with regard to the precision metric.

Similar hypotheses were tested with respect to the I-EQM classifier and the recall metric. A significance level of $\alpha = 0.05$ was employed for all conducted tests. Thus, the tests reject the null hypothesis H_0 if a p-value smaller than α is obtained; otherwise the null hypothesis is accepted. Table XII presents the corresponding findings. The first column of the table refers to the considered effectiveness measures, the second one to the classification methods being compared and the last one describes the obtained p-values (two-tailed). With respect to precision, on the one hand, the *null hypothesis* is rejected only in the case of 'HOM versus I-EQM', i.e., regarding the precision metric, there is a statistically significant difference in the effectiveness of the two classifiers. For the remaining cases, i.e., 'CI versus HOM' and 'CI versus I-EQM', the null hypothesis is accepted. On the other hand, with respect to the recall measure, the null hypothesis is rejected in the cases of 'CI versus I-EQM' and 'HOM versus I-EQM', thus, it can be concluded that there is strong evidence to establish a difference in the effectiveness of the I-EQM classifier with respect to the HOM and coverage impact methods. Finally, in the case of 'CI versus HOM', the null hypothesis is accepted. These findings suggest that there is statistically insufficient evidence to establish a difference in the performance of the coverage impact and I-EQM classifiers concerning precision, on the contrary, there is a strong statistically significant difference in their performance regarding recall, with a p-value of 0.016, as per Table XII.

5.6. Discussion

Comparing the coverage impact method's results between the two sets, a slight increase in the precision metric of the control mutant set 2 and a minor decrease in the corresponding recall value can be observed. These variations indicate that the coverage impact's classification ability is not greatly affected by different mutants. Considering the classification results of the HOM classifier's, the aforementioned trend is also present, that is, the precision of the control mutant set 2 is increased while its recall is decreased. It must be mentioned that in this case, the deviation is higher, indicating that HOM's classification ability is affected more by different mutants than the one of the coverage impact technique. Finally, the trend observed for the previous classifiers is consistent with the differences between the results of the I-EQM technique, for the two control mutant sets. Concerning its results, it is obvious that the I-EQM classifier manages to enhance the recall of the coverage impact technique for both the examined control mutant sets (Table VIII and Table IX) while maintaining its precision at a reasonably high level.

Another aspect of the presented results that should be noted relates to the HOM classifier (same method foms) approach. By examining the entries of Tables VI and VII, it is apparent that this approach experiences a decrease of nearly 10% on its recall value compared with the other variations of the HOM classifier (i.e., killed foms and all foms) and approximately 20% compared with the one of the coverage impact method. Interestingly, the I-EQM (same method foms) classification scheme, according to Table XI, suffers only a loss of 3% compared with the rest of the approaches

DOI: 10.1002/stvr

of the I-EQM classifier and has an enhanced recall of nearly 15% compared with the coverage impact classifier. These findings suggest that the majority of the misclassified killable mutants of the coverage impact classifier can be correctly classified by the HOM classifier (same method foms) variation.

Generally, the application of the I-EQM classifier on the studied subjects results in identifying the 81% of the live killable mutants and 46% of the total instances of equivalent mutants. The coverage impact classifier identifies the 65% of the live killable mutants and 33% of the equivalent ones. These values suggest that by killing the identified killable mutants, a mutation score[¶¶] such as 95.2% will be achieved for the coverage impact technique and one of 97.4% for the I-EQM classifier. Therefore, it becomes obvious that a more thorough testing process is established by the I-EQM method. However, this is borne with the overhead of analysing 1.17% more equivalent mutants.[|||]

6. MUTANTS WITH HIGHER IMPACT

The behaviour of the considered approaches with respect to higher impact values is also investigated. To this end, the impact values of the examined mutants based on the coverage impact and the HOM classifiers are determined for both examined control mutant sets. Note that the impact value of a mutant based on the coverage impact technique is defined as 'the number of methods that have at least one statement that is executed at a different frequency in the mutated run than in the normal run, while leaving out the method that contains the mutation' [19]. The corresponding impact value of the HOM classifier is defined as the number of first-order mutants in the CM set whose output has changed after being combined with the live-unclassified mutant (Section 3.D).

Figure 10 presents the results of the coverage impact technique for both examined control mutant sets. The left part of the figure provides information about the precision metric (y-axis) and how it changes according to different impact value thresholds (x-axis). Based on different thresholds, different mutants are classified as possibly killable or possibly equivalent. For instance, by employing a threshold value of 50, mutants with higher impact values will be classified as possibly killable, otherwise they will be classified as possibly equivalent. From the left part of the figure, it can be seen that for impact values less than 20, the precision metric increases, whereas for impact values between 20 and 100, the opposite holds. Note that the highest precision scores are obtained when the impact value thresholds are between 10 and 20. The right part of the figure describes a plot between precision (y-axis) and the percentage of mutants with the highest impact (x-axis); for example, an x_1 value of 0.2 indicates that the 20% of these mutants are examined. Thus, lower values of the x-axis represent mutants with higher impact values. It can be observed that, in general, the precision metric decreases as the percentage of the examined mutants with the highest impact decreases. Consider the top 10% of the mutants with the highest impact (i.e., $x_1 = 0.1$), by examining only these mutants a precision value of approximately 70% is obtained. In contrast, by examining the top 30%, a precision score of 80% is realised. These findings suggest that an analogy between mutants with the highest impact and high killability ratios cannot be drawn.

Figure 11 depicts the same results as Figure 10 but this time for the HOM classifier. Examining the two figures, it is apparent that the same trends exist.

In conclusion, it can be argued that mutants with the highest impact do not necessarily guarantee the highest killability ratios. On the contrary, the results suggest that the precision metric for higher impact values decreases. This trend is in accordance with the findings of Schuler and Zeller [19, 20], where the precision of the top 25% of the mutants with the highest impact was found higher than the one obtained by examining only the top 15% (Table IX [19]). The presented results indicate that the relation between impact and killability needs further investigation.

[¶¶] These scores were evaluated by counting the mutants killed by the employed tests plus the identified killable mutants of the totally estimated killable ones.

[|||] This number was evaluated by counting the identified equivalent mutants of the totally estimated number of equivalent ones.

DOI: 10.1002/stvr

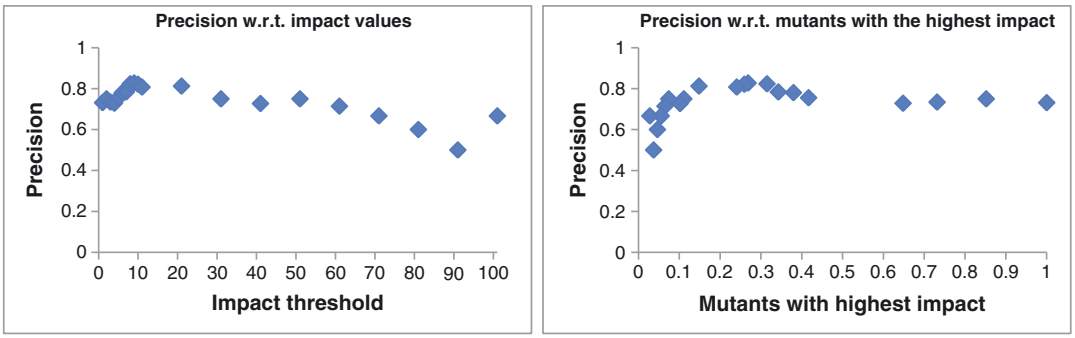

Figure 10. Coverage impact classifier: Precision with regard to Impact values (left) and Mutants with the highest impact (right).

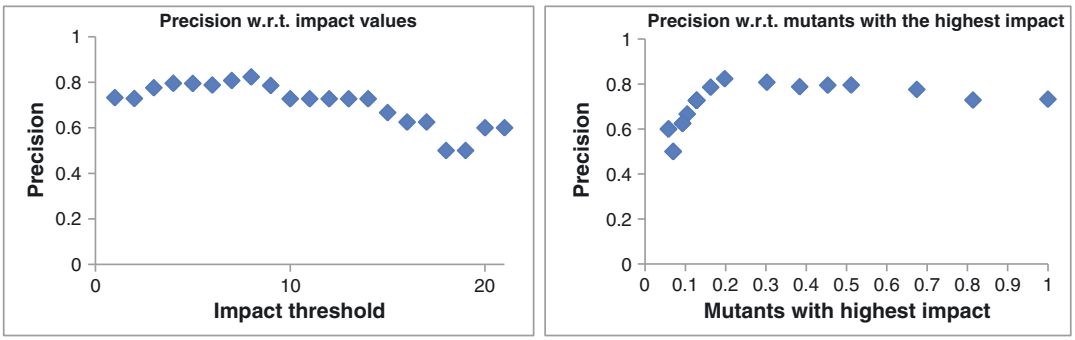

Figure 11. HOM classifier: Precision with regard to Impact values (left) and Mutants with the highest impact (right).

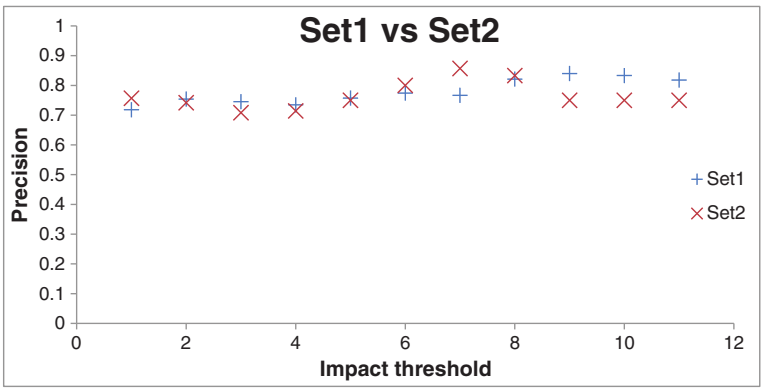

Figure 12. Coverage impact classifier: Precision with regard to Impact values for control mutant set 1 and 2.

To investigate the differences between the control mutant sets 1 and 2, Figure 12 displays the coverage impact's classification precision (y-axis) with respect to the different impact value thresholds (x-axis). It must be noted that the maximum impact value considered for this diagram is 12, because it is the maximum impact value of the control mutant set 2. From this graph, it can be observed that for impact values of less than 8, both sets demonstrate a similar behaviour. For impact values greater than 8, the precision of the control mutant set 1 remains approximately the same, whereas the one of the control mutant set 2 decreases. Overall, because the differences between the two sets are small, it is believed that the above conclusion holds for both studied sets.

DOI: 10.1002/stvr

7. THREATS TO VALIDITY

This section discusses possible threats to the validity of the findings of the present study.

The *internal validity* concerns the degree of confidence in the causal relationship of the studied factors and the observed results. One such factor is the utilization of the specified test subjects and the use of the Javalanche framework. As mentioned before, their choice was mainly based on enabling a direct comparison between the proposed classification techniques and the coverage impact classifier. Because the employed subjects are large in size and of different application domains, they are considered as appropriate. Furthermore, many recent empirical studies [17, 19, 20, 22] utilised Javalanche, thus increasing the confidence in its results.

The employed mutant operators constitute another possible issue. Different operator sets, as those suggested by Offutt *et al.* [10], might give different classification results. However, the present study focuses on the mutants' impact, which is believed to be an attribute independent of the nature of the studied mutants [20]. Additionally, the present study replicates the findings of the coverage impact technique [19, 20], thus it is natural to use the same mutants.

Another potential threat that falls into this category concerns the employed test suites. It is possible that different test suites could produce different classification results. However, these tests were independently created by the developers of the employed programs without using mutation testing. Further, the manually classified mutants were randomly chosen among those that were executed and not killed by the employed tests. These two facts give confidence that the studied methods can provide useful guidance in increasing the quality of the testing process.

The *external validity* of an experiment refers to the potential threats that inhibit the generalisation of its results. The generalisation of a study's results is difficult to be achieved because of the range of different aspects that the experimental study must consider. The results presented in this paper are no exception, though effort has been made to attenuate the effects of the aforementioned threats. First, the empirical evaluation of this study was based on a benchmark set of real-world programs, which has been used in similar research studies [17, 19, 20, 22]. Second, the examined programs vary in size and complexity. Finally, the evaluation of the proposed classification techniques was based on two independently created sets of manually classified mutations, one created for the evaluation of the coverage impact classifier [19, 20] and the other for the purposes of the present experimental study.

The *construct validity* refers to the means of defining the employed measurement of an experiment and the extent to which it measures the intended properties. A possible threat is relevant to the manual classification of the mutants of the examined control mutant sets. The mutants classified as non-equivalent pose no threat as the basis of their classification is a test case able to 'kill' them, whereas the mutants classified as equivalent could do so due to the complexity of the involved manual analysis.

8. RELATED WORK

Dynamically isolating possible equivalent mutants with the aim of reducing their effects in the testing process is a relatively new direction of the mutation testing research. One of the first attempts to tackle this issue is that due to Adamopoulos *et al.* [31]. In their work, genetic algorithms were employed in order to overcome the difficulties of the large numbers of both the introduced and the equivalent mutants. Their results suggested that it is possible to generate a small set of killable mutants, valuable to the testing process. However, contrary to the present approach, this technique attempts to produce killable mutants and not to isolate equivalent ones in order to help assessing the test adequacy.

Measuring the mutants' impact as the means to assess mutations has been proposed by Schuler *et al.* [22] with the use of dynamic program invariants. In their study, it was found that when mutants break dynamically generated invariants, they are more likely to be killable. The use of coverage impact as an assessment measure of mutants was initially suggested by Grun *et al.* [21] and later extended by Schuler and Zeller [19, 20]. Empirical evaluation [19, 20] of the aforementioned approaches based on the Javalanche [11] tool suggested that coverage impact is more efficient and

DOI: 10.1002/stvr

effective in classifying killable mutants. Therefore, the *I-EQM* approach, proposed in the present paper, uses and extends the coverage impact method with the aim of ameliorating the method's recall, that is, retrieving most of the killable mutants. Details about the coverage impact approach have been given in Section 3.

Heuristics able to detect some equivalent mutants do exist [12, 32, 33]. In such an attempt, Baldwin and Seyward [32] suggested the use of compiler optimization techniques. Because program optimization produces equivalent program versions, the original and the mutant programs can be optimised or de-optimised and eventually identify equivalent mutants. Offutt and Craft [28] developed such techniques and based on their empirical evaluation found that a 10% of the existing equivalent mutants could be automatically detected. This approach was later extended by Offutt and Pan [12] using path analysis and a constraint-based testing technique. In their evaluation, Offutt and Pan report that they achieved to detect the 45% of the existing equivalent mutants on average. Other related approaches, make use of program slicing [34] or dependence analysis [35] to aid the tester in identifying equivalent mutants. All the equivalent mutant detection techniques, mentioned thus far, can be applied complementary to the mutant classification approaches proposed in the present paper. Thus, these approaches can identify equivalent mutants and then I-EQM can be applied to the remaining uncategorised mutants.

Mutation testing utilizing higher-order mutants has been studied by Jia and Harman [26] who suggested that the use of higher-order mutants could be profitable. In view of this, they propose the use of search-based optimization techniques in order to construct hard-to-kill, higher-order mutants. To this end, practical problems of mutation testing could be answered by using only such higher-order mutants. In a follow up work [36, 37] in order to better simulate real faults, higher-order mutants were constructed based on a multi objective evolutionary method. For this reason, mutants that were both hard-to-kill and syntactically similar to the original program were produced. Further, automated tools that enable testing based on higher-order mutants [38, 39] have also been suggested. A synopsis on higher-order mutation and search-based testing can be found in the work of Harman *et al.* [40].

Second-order mutation has been addressed in the literature as an alternative method to first-order mutation [9, 28]. According to these approaches, equivalent mutants are considerably less likely to occur. Thus, their effects on the testing process are reduced. Generally, the higher-order mutation approaches that appeared in the literature, try to produce mutant sets with fewer equivalent mutants and not to isolate them. The approach presented in this paper is the first one, to the authors' knowledge, that uses higher-order mutants in order to identify likely to be first-order equivalent mutants.

9. CONCLUSION

The practical application of mutation testing constitutes the primary aim of the present work. Towards this direction, a dynamic method, I-EQM, able to isolate possible equivalent mutants was suggested. The originality of the proposed technique stems from the fact that it leverages higher-order mutation and the mutants' coverage impact. Specifically, the classification scheme utilises second-order mutation to classify a given set of first-order mutants as possibly killable or not. I-EQM extends previously proposed approaches [19, 20] with the aim of correctly classifying a significantly greater number of killable mutants.

In summary, this paper presents: (a) a novel dynamic method based on higher-order mutation, able to classify first-order mutants as possibly killable or possibly equivalent. Additionally, (b) an empirical study based on a manually classified mutant set replicating the coverage impact technique. Furthermore, (c) empirical results, which reveal that I-EQM, achieves the correct classification of 81% of the killable mutants with a precision of 71%. This particular claim is also supported by the statistical analysis, performed on the obtained results, which reinforces the argument that I-EQM provides a better classification scheme than the other methods. Finally, (d) a manually identified and publicly available set of killable and equivalent mutants. This set amplifies an existing benchmark set [19, 20] with the aim of enabling further studies. Such being the case, the first mutant set [19, 20] can be used to develop techniques and the second one can be used to evaluate them.

DOI: 10.1002/stvr

The research presented in this paper seeks practical solutions to the application of mutation testing. It can be extended by using a selective set of mutants. Such a set could provide even better and computationally inexpensive classification results. Additionally, it can be used to classify higher-order mutants, that is, n order mutants can be classified by using $n + 1$ order mutation testing. Further directions include conducting additional experiments by employing different mutant sets and subject programs to revalidate the findings of the present paper.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments that helped improve the quality of the present paper.

REFERENCES

1. Offutt AJ, Untch RH. Mutation 2000: uniting the orthogonal. In *Mutation Testing for the New Century*, Wong WE (ed.) Kluwer Academic Publishers: Norwell, MA, USA, 2001; 34–44.
2. Mathur AP, Wong WE. An empirical comparison of data flow and mutation-based test adequacy criteria. *Software Testing, Verification and Reliability* 1994; **4**(1):9–31.
3. Offutt AJ, Pan J, Tewary K, Zhang T. An experimental evaluation of data flow and mutation testing. *Software: Practice and Experience* 1996; **26**(2):165–176.
4. Ammann P, Offutt J. *Introduction to Software Testing*. Cambridge University Press: Cambridge, UK, 2008.
5. Li N, Praphamontripong U, Offutt AJ. An experimental comparison of four unit test criteria: mutation, edge-pair, all-uses and prime path coverage. *Proceedings of the 4th International Workshop on Mutation Analysis (MUTATION'09),* Denver, CO, USA, 2009; 220–229.
6. Andrews JH, Briand LC, Labiche Y. Is mutation an appropriate tool for testing experiments? *Proceedings of the 27th International Conference on Software Engineering,* St. Louis, Missouri, USA, 2005; 402–411.
7. Andrews JH, Briand LC, Labiche Y, Namin AS. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Transactions on Software Engineering* 2006; **32**(8):608–624.
8. Jia Y, Harman M. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering* 2011; **37**(5):649–678.
9. Papadakis M, Malevris N. An empirical evaluation of the first and second order mutation testing strategies. *2010 Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW),* Paris, France, 2010; 90–99.
10. Offutt AJ, Lee A, Rothermel G, Untch RH, Zapf C. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering Methodology* 1996; **5**(2):99–118.
11. Schuler D, Zeller A. Javalanche: Efficient mutation testing for Java. *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering,* Amsterdam, The Netherlands, 2009; 297–298.
12. Offutt AJ, Pan J. Automatically detecting equivalent mutants and infeasible paths. *Software Testing, Verification and Reliability* 1997; **7**:165–192.
13. Budd TA, Angluin D. Two notions of correctness and their relation to testing. *Acta Informatica* 1982; **18**(1):31–45.
14. Papadakis M, Malevris N. Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing. *Software Quality Journal* 2011; **19**(4):691–723.
15. Papadakis M, Malevris N. Automatic mutation test case generation via dynamic symbolic execution. *2010 IEEE 21st International Symposium on Software Reliability Engineering (ISSRE),* San Jose, California, USA, 2010; 121–130.
16. Papadakis M, Malevris N. Mutation based test case generation via a path selection strategy. *Information and Software Technology* 2012; **54**(9):915–932.
17. Fraser G, Zeller A. Mutation-driven generation of unit tests and oracles. *Proceedings of the 19th International Symposium on Software Testing and Analysis,* Trento, Italy, 2010; 147–158.
18. Baudry B, Fleurey F, Jézéquel J-M, Traon YL. From genetic to bacteriological algorithms for mutation-based testing: Research articles. *Software Testing, Verification and Reliability* 2005; **15**(2):73–96.
19. Schuler D, Zeller A. (Un-)Covering equivalent mutants. *2010 Third International Conference on Software Testing, Verification and Validation (ICST),* Paris, France, 2010; 45–54.
20. Schuler D, Zeller A. Covering and uncovering equivalent mutants. *Software Testing, Verification and Reliability* 2013; **23**(5):353–374.
21. Grun BJM, Schuler D, Zeller A. The impact of equivalent mutants. *International Conference on Software Testing, Verification and Validation Workshops, 2009. ICSTW'09,* Denver, CO, USA, 2009; 192–199.
22. Schuler D, Dallmeier V, Zeller A. Efficient mutation testing by checking invariant violations. *Proceedings of the eighteenth international symposium on Software testing and analysis,* ACM: Chicago, IL, USA, 2009; 69–80.
23. DeMillo RA, Lipton RJ, Sayward FG. Hints on test data selection: Help for the practicing programmer. *Computer* 1978; **11**(4):34–41.

DOI: 10.1002/stvr

24. Offutt AJ. Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering Methodology* 1992; **1**(1):5–20.
25. Jia Y, Harman M. Constructing subtle faults using higher order mutation testing. *Proceedings of the 8th International Working Conference on Source Code Analysis and Manipulation (SCAM'08)*: Beijing, China, 2008; 249–258.
26. Jia Y, Harman M. Higher order mutation testing. *Information and Software Technology* 2009; **51**(10):1379–1393.
27. Polo M, Piattini M, García-Rodríguez I. Decreasing the cost of mutation testing with second-order mutants. *Software Testing, Verification and Reliability* 2009; **19**(2):111–131.
28. Kintis M, Papadakis M, Malevris N. Evaluating mutation testing alternatives: A collateral experiment. *Software Engineering Conference (APSEC), 2010 17th Asia Pacific,* Sydney, Australia, 2010; 300–309.
29. Kintis M, Papadakis M, Malevris N. Isolating first order equivalent mutants via second order mutation. *5th International Conference on Software Testing, Verification and Validation,* Montreal, Quebec Canada, 2012; 701–710.
30. Manning CD, Raghavan P, Schütze H. *Introduction to Information Retrieval*. Cambridge University Press: Cambridge, UK, 2008.
31. Adamopoulos K, Harman M, Hierons R.M. How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution. In *GECCO (2)*, Vol. 3103, Kalyanmoy D (ed.) Springer: Berlin Heidelberg, 2004; 1338–1349.
32. Baldwin D, Sayward FG. *Heuristics for Determining Equivalence of Program Mutations*. Yale University: New Haven, Connecticut, 1979.
33. Offutt AJ, Craft WM. Using compiler optimization techniques to detect equivalent mutants. *Software Testing, Verification and Reliability* 1994; **4**(3):131–154.
34. Hierons RM, Harman M, Danicic S. Using program slicing to assist in the detection of equivalent mutants. *Software Testing, Verification and Reliability* 1999; **9**(4):233–262.
35. Harman M, Hierons RM, Danicic S. The relationship between program dependence and mutation analysis. *Proceedings of the 1st Workshop on Mutation Analysis (MUTATION'00)*: San Jose, California, 2001; 5–13.
36. Langdon WB, Harman M, Jia Y. Efficient multi-objective higher order mutation testing with genetic programming. *Journal of Systems and Software* 2010; **83**(12):2416–2430.
37. Langdon WB, Harman M, Jia Y. Multi objective higher order mutation testing with genetic programming. *Proceedings of the 2009 Testing: Academic and Industrial Conference - Practice and Research Techniques,* IEEE Computer Society: Windsor, UK, 2009; 21–29.
38. Yue J, Harman M. MILU: A customizable, runtime-optimized higher order mutation testing tool for the full C language. *Practice and Research Techniques, 2008. TAIC PART'08, Testing: Academic & Industrial Conference*: Windsor, UK, 2008; 94–98.
39. Harman M, Jia Y, Langdon WB. Strong higher order mutation-based test data generation. *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*: ACM, Szeged, Hungary, 2011; 212–222.
40. Harman M, Jia Y, Langdon WB. A manifesto for higher order mutation testing. *Proceedings of the 5th International Workshop on Mutation Analysis (MUTATION'10)*: Paris, France, 2010; 80–89.

DOI: 10.1002/stvr