

Employing Second Order Mutation for Isolating First Order Equivalent Mutants

Marinos Kintis¹, Mike Papadakis² and Nicos Malevris³

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 utilized 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.

Keywords; *Mutation testing; Equivalent mutants; Higher Order mutation; Mutants' Impact; Dynamic analysis*

I. 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. Since 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 behavior is explored. Thus, the testing process attempts to find an adequate test set that forces the software to exhibit a wide range of possible behaviors. 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 i.e., 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*, i.e., 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 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 behavior to real faults [6], [7]. Thus, if mutants can be killed, then programming faults might also be exposed. Although

¹ Athens University of Economics and Business, Athens, Greece, Email: kintism@aueb.gr

² Interdisciplinary Center for Security, Reliability and Trust, Luxembourg University, Email: michail.papadakis@uni.lu

³ Athens University of Economics and Business, Athens, Greece, Email: ngm@aueb.gr

powerful, mutation is not widely used for testing. This may be due to its excessive computational demands, i.e., 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], utilized 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, reduced 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 abovementioned problems, the automated production of test cases and the adequacy evaluation i.e., 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 don’t 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 was comprised of 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, losing a high number of killable mutants. The approach described by the present paper expands the *coverage impact* method by aiming at overcoming the abovementioned 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, i.e., 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 utilizes both abovementioned techniques with promising results.

The present work describes three variations of the proposed technique and empirically investigates their ability to classify killable mutants. Additionally, it empirically validates the *coverage impact* method and compares it with the proposed ones. Empirical 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 retrieving approximately 20% more killable mutants. In general, the paper’s contributions can be summarized by the following three points:

- 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%, while the proposed technique realizes 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 available⁴ with the aim of enabling replication and comparative studies.

The rest of the paper is organized as follows: Sections II and III introduce the core concepts and describe the studied mutant classification techniques. Sections IV and V present the utilized benchmarks and the experimental results. Sections VI and VII discuss issues regarding the mutants' impact and possible threats to validity. Section VIII refers to related work and in Section IX conclusions and possible future directions are given.

II. 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.

A. 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 to 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 mutants) 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 to be an "undecidable" problem in general [13]. Additionally, identifying manually one equivalent mutant requires approximately 15 minutes [19], [20]. Since 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 III.

⁴ This set and the results of the present study are publicly available at: <http://pages.cs.aueb.gr/~kintism/#stvr2012>

B. 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 ones 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 [27], [28] it was shown that substantial benefits can be gained by using *soms* instead of *foms*. 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 behavior of the second order ones.

C. 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 some 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 this end, 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 behavior 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 abovementioned approaches can be classified appropriately.

D. 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 the study on the equivalent mutant classification [19], [20] were used. Thus, the present study utilizes 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 such as [17], [19-22]. Table I records details about the utilized mutant operators supported by the tool and the present study.

III. MUTANT CLASSIFICATION

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

A. Mutation analysis using mutant classifiers

Applying first order mutation testing entails the generation of a mutant set, referred to as the candidate mutant set. Then, 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 analyzed 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 while the number of killable ones decreases (since they are killed). Thus, if the live mutants could be automatically classified as killable and equivalent, one could claim substantial benefits by analyzing only the killable ones [19], [20], [22], [29]. In such a situation, 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.

TABLE I. MUTANT OPERATORS UTILIZED BY JAVALANCHE

Mutant Operators	Description
replace numerical constant	Replaces a numerical constant instance by a value + 1, a value -1 or by 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).

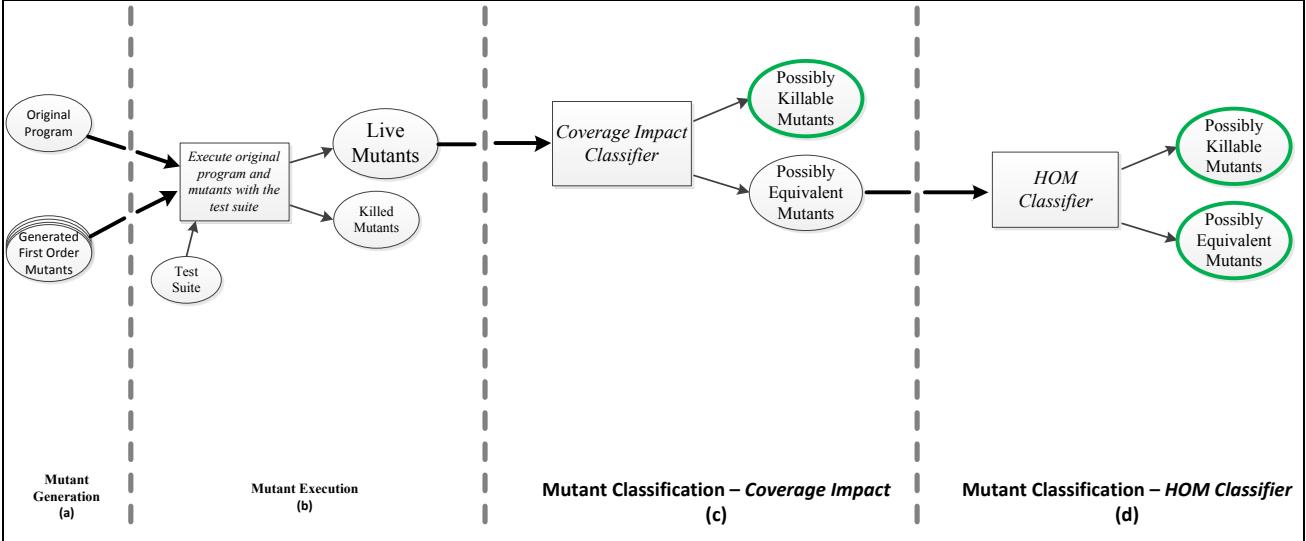


Figure 1. Mutant Classification Process. The live mutants are classified as Possibly Killable and Possibly Equivalent. The *I-EQM* process works in two phases. First, the live mutants will be classified via the *Coverage Impact Classifier* and subsequently the produced Possibly Equivalent Mutant set will be classified by the *HOM Classifier*. The highlighted first order mutant sets are the outcome of the *I-EQM* classification scheme.

Based on the above 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 mutants are then categorized as possibly killable and possibly equivalent ones (Figure 1 (c)). The testing process is then continued based on the possibly killable mutant set.

B. Mutant classification effectiveness measures

Mutant classification categorizes mutants as possibly killable and possibly equivalent ones. Being a heuristic method, mutant classification may correctly classify some mutants, e.g. a killable mutant correctly classified as possibly killable and fail on others, e.g. an equivalent mutant 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 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 utilized. These metrics are usually employed in order to compare classifiers in information retrieval experiments [30].

$$\text{Precision} = \frac{\text{True Killable}}{\text{True Killable} + \text{False Killable}}$$

$$\text{Recall} = \frac{\text{True Killable}}{\text{True Killable} + \text{False Equivalent}}$$

$$\text{Accuracy} = \frac{\text{True Killable} + \text{True Equivalent}}{\text{True Killable} + \text{False Killable} + \text{True Equivalent} + \text{False Equivalent}}$$

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

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 categorize 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, while 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 score⁵ constitutes a general metric which 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 V.

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, i.e., by the recall metric, since 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, since these mutants will be manually analyzed by the tester. Thus, the methods' efficiency can be expressed by the precision⁶ value of the classifier.

In general, high precision is difficult to be achieved, but 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. Since 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 categorize many equivalent mutants as possibly killable ones (a case

⁵ For non-negative real values of β .

⁶ The percentage of the equivalent mutants that are incorrectly classified as possibly killable is actually $(1 - \text{precision})$. Thus, higher precision indicates fewer equivalent mutants to be considered and hence, a higher efficiency.

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.

C. 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*.

D. 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 *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.

HOM Classifier categorizes mutants based on the impact they have on each other. In view of this, it produces pairs of mutants by combining the examined (first order) mutant with others. The classifier works based on the intuition that since 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.

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,

$$\text{outputOf}(\textit{fom}, \textit{test}) \neq \text{outputOf}(\textit{umut.fom}, \textit{test}) \Rightarrow \textit{umut} \in \text{Killable}$$

The above hypothesis forms the basis of the proposed classification scheme. Thus, if the condition of the above 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.

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 categorized as possibly killable or possibly equivalent based on the truth or the falsehood of the classification's predicate.

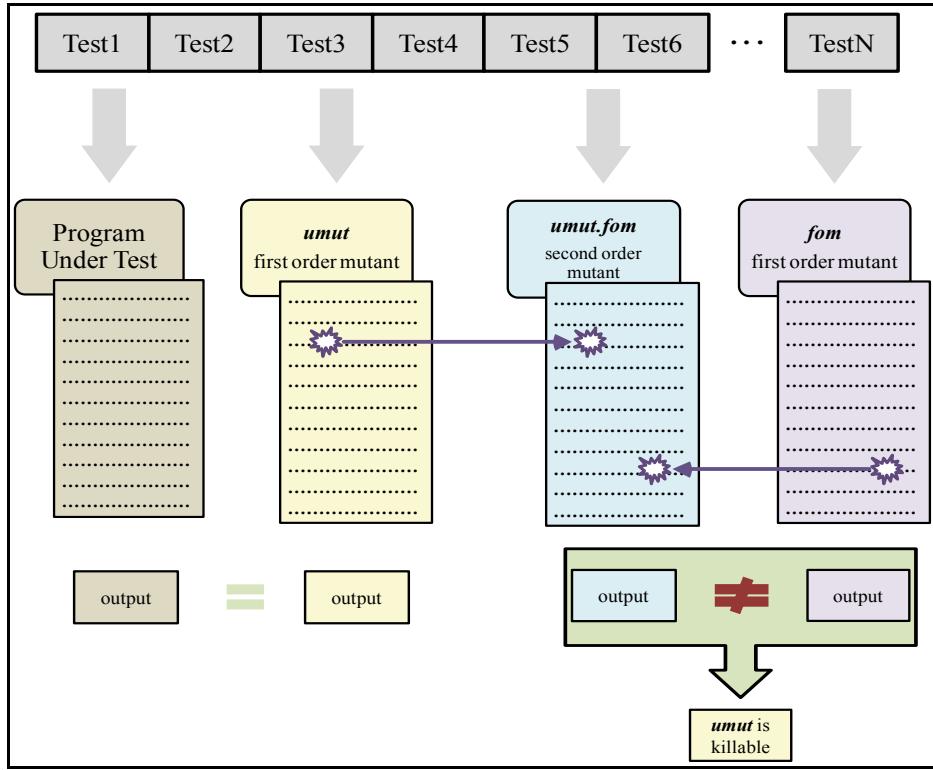


Figure 2. The unclassified first order mutant *umut* is classified as possibly killable based on its impact on other *fom* mutants.

- 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 abovementioned 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 classification mutant 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 classification mutant (CM) set whose output is changed after being combined with the live-unclassified mutant.

3) Classification Mutant (CM) set

Generally, the proposed classification technique determines the impact of mutants using other mutations referred to as the classification mutant (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 get the best results. However, such an approach is prohibitive, since it requires executing all mutants with all tests for each mutant to be classified.

Considering the above issue a necessary restriction was imposed on the utilized 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

```

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 outputs of fom and umut,fom
10.      if the outputs 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.

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 since 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.

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 utilize only a specific subset. The *HOM Classifier (killed foms)* uses only those mutants that have been killed by the employed tests while 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 since these mutants are more sensitive to the utilized 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 above, 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 due to the reduced number of mutant combinations and their respective executions.

E. 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 *Isolating Equivalent Mutants (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 classify correctly most of the mutants that are correctly classified by the rest of the examined approaches [19], [20]. Since the corresponding precision measure of the *Coverage Impact Classifier* can be considered as an accurate one, i.e., 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 summarized 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, i.e., 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 categorized as such by at least one of the combined classifiers. The possibly equivalent mutant set will contain the mutants that are categorized as such by all the combined classifiers. This can be also generalized to a classification process that includes more than two classifiers.

IV. 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 utilized in various mutation testing studies, such as [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 the “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

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.

the present paper in a similar fashion to the previous respective 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 utilized 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 executed by the available test cases. Finally, the “Manually Classified Mutations” 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 while 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. Since 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

TABLE III. MUTANT SAMPLES PROFILE W.R.T. SUBJECT PROGRAM

Subject Programs	Considered Mutants		Manually Classified Mutants					
	Number of Mutants	Reached Mutants	Killable			Equivalent		
			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%)

TABLE IV. MUTANT SAMPLES PROFILE W.R.T. 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%)

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.

V. 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 III. 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.

A. Research Objectives

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

- *Can second order mutation provide adequate guidance in equivalent mutant isolation, i.e., 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?*

B. 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 experiment⁷ 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 cases⁸ 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 seconds for timeout limit⁹ and b) execution of

⁷ 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).

⁸ The same tests as the previous studies [19], [20] were employed.

⁹ To avoid discrepancies caused by the introduced mutants, such as endless loops, a timeout limit was set. If mutant execution time exceeds this limit, the mutant is treated as killed.

all tests with all mutants. Since Javalanche does not support second order mutation, the applied process for generating second order mutants is presented in Figure 4. Initially, the source 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 position¹⁰ as the examined one and were 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, i.e., 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 to classify the examined mutants.

The above process was performed for the *HOM* and *I-EQM Classifiers* for all their respective variants i.e., *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 *I-EQM Classifier*, presented in Figure 1, the *Coverage Impact Classifier* classifies the examined mutants as possibly killable and possibly equivalent ones; the mutants classified as possibly equivalent are subsequently categorized 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 utilized 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 realized by evaluating the F_β measure with $\beta = 1$. The second scenario emphasizes 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.

```

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
5.       produced by the same mutant operator
6.         remove umut,fom from the resulting set of second order mutants
7.     end if
8.   end for each
9. return the resulting set of second order mutants
end for each
```

Figure 4. Generation Process of Second Order Mutants.

¹⁰ Position refers to the exact part of the original source code that differs from the mutant programs.

In the present experiment, special care was taken to handle certain cases due to some inconsistencies of the utilized 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 categorized as possibly killable due to timeout conditions (if either the first order mutant or the second order one results in a timeout) the corresponding mutants of the considered mutant 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 due to the different 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. Joda-Time and AspectJ test subjects, or folder locations, e.g. the JTopas program, are some 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.

C. 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 sub-columns 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 them and the ones reported in the respective studies of Schuler and Zeller [19], [20] should be 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 V. MUTANT CLASSIFICATION USING THE *COVERAGE IMPACT* CLASSIFIER

Subject Programs	Killable Set1		Equivalent Set1		Killable Set2		Equivalent Set2		Possibly killable Set1		Possibly killable Set2	
	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%

TABLE VI. MUTANT CLASSIFICATION USING THE *HOM* CLASSIFIER – CONTROL MUTANT SET 1
(DENOTING: ALL FOMS – KILLED FOMS – SAME METHOD FOMS)

Subject Programs	Killable Set1			Equivalent Set1			Possibly killable Set1		
	True Killable	False Equivalent	True Equivalent	False Killable	Precision %	Recall %			
ASPECTJ	8	8	4	7	7	11	2	2	4
BARBECUE	9	7	7	5	7	7	5	5	5
COMMONS	4	4	4	2	2	2	10	10	12
JAXEN	6	6	3	4	4	7	7	8	3
JODA-TIME	9	9	9	5	5	5	4	5	5
JTOPAS	3	3	3	7	7	7	10	10	10
XSTREAM	5	5	5	3	3	3	5	6	5
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
(DENOTING: ALL FOMS – KILLED FOMS – SAME METHOD FOMS)

Subject Programs	Killable Set2			Equivalent Set2			Possibly killable Set2		
	True Killable	False Equivalent	True Equivalent	False Killable	Precision %	Recall %			
ASPECTJ	5	5	4	1	1	2	3	3	3
BARBECUE	2	2	2	7	7	7	1	1	1
COMMONS	3	3	2	3	3	4	3	3	3
JAXEN	4	4	3	3	3	4	3	3	3
JODA-TIME	4	4	3	3	3	4	3	3	3
JTOPAS	1	0	0	3	4	4	6	6	6
XSTREAM	3	3	2	2	2	3	4	4	4
TOTAL	22	21	16	22	23	28	23	23	23
							3	3	3
							88	88	84
							50	48	36

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 realizes a precision of 70% and a recall of 55% and the *HOM Classifier (same method foms)* 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% respectively¹¹. Finally, with respect to the second control mutant set, the *I-EQM Classifier*'s variation that utilizes the *HOM Classifier (all foms)* technique realizes 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%.

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 classification mutant set (CM set) achieves approximately the same classification effectiveness as the employment of

¹¹ 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 V.B.).

TABLE VIII. MUTANT CLASSIFICATION USING THE *I-EQM* CLASSIFIER – CONTROL MUTANT SET 1
(DENOTING: ALL FOMS – KILLED FOMS – SAME METHOD FOMS)

Subject Programs	Killable Set1			Equivalent Set1			Possibly killable Set1		
	True Killable	False Equivalent	True Equivalent	False Killable	Precision %	Recall %			
ASPECTJ	14	14	13	1	1	2	0	0	0
BARBECUE	12	12	12	2	2	2	2	4	4
COMMONS	4	4	4	2	2	2	8	8	10
JAXEN	6	6	5	4	4	5	6	6	7
JODA-TIME	13	13	13	1	1	1	2	3	3
JTOPAS	9	9	9	1	1	1	10	10	10
XSTREAM	6	6	6	2	2	2	3	4	3
TOTAL	64	64	62	13	13	15	31	33	35
							32	30	28
							67	68	69
								83	83
									81

TABLE IX. MUTANT CLASSIFICATION USING THE *I-EQM* CLASSIFIER – CONTROL MUTANT SET 2
(DENOTING: ALL FOMS – KILLED FOMS – SAME METHOD FOMS)

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

all the generated first order mutants. Concerning the *HOM Classifier (same method foms)* technique, it is less effective than the other two variations, but in most cases is more efficient due to the reduced size of the considered classification mutant set (CM set). With these facts evident, since the *HOM Classifier (killed foms)* is more efficient than *HOM Classifier (all foms)* its use is advisable. Finally, compared to the *Coverage Impact Classifier*, the *HOM Classifier* attains lower precision and recall values, indicating that the *Coverage Impact Classifier* is a better one.

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 and Table XI presents 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 foms)* approach realizes 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 utilizes the *HOM Classifier (same method foms)* achieves a precision of 71% and a recall value of 79%. These results indicate that all three variations realize 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

TABLE X. MUTANT CLASSIFICATION USING *COVERAGE IMPACT* CLASSIFIER ON BOTH SAMPLES

Subject Programs	Killable Set1+Set2		Equivalent Set1+Set2		Possibly killable Set1+Set2	
	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 *I-EQM* CLASSIFIER ON BOTH SAMPLES

(DENOTING: ALL FOMS – KILLED FOMS – SAME METHOD FOMS)

Subject Programs	Killable Set1+Set2			Equivalent Set1+Set2			Possibly killable Set1+Set2		
	True Killable	False Equiv	True Equiv	False Killable	Precision %	Recall %			
ASPECTJ	20	20	19	1	1	2	0	0	0
BARBECUE	20	20	20	3	3	3	3	4	4
COMMONS	7	7	6	5	5	6	11	11	13
JAXEN	11	11	10	6	6	7	9	9	10
JODA-TIME	18	18	18	3	3	3	2	3	3
JTOPAS	12	11	11	2	3	3	16	16	16
XSTREAM	11	11	11	2	2	2	5	6	5
TOTAL	99	98	95	22	23	26	46	48	50
				43	41	39	70	71	71
							82	81	79

of its precision. In particular, the *I-EQM* technique realizes 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 categorizes 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 method. 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 V, 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, while 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 performs better than the rest. The second scenario, described by the F_2 measure (β equal to 2), emphasizes on the recall value. It 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, since the *ALL Mutants* categorizes all examined mutants as possibly killable. Finally, the last scenario,

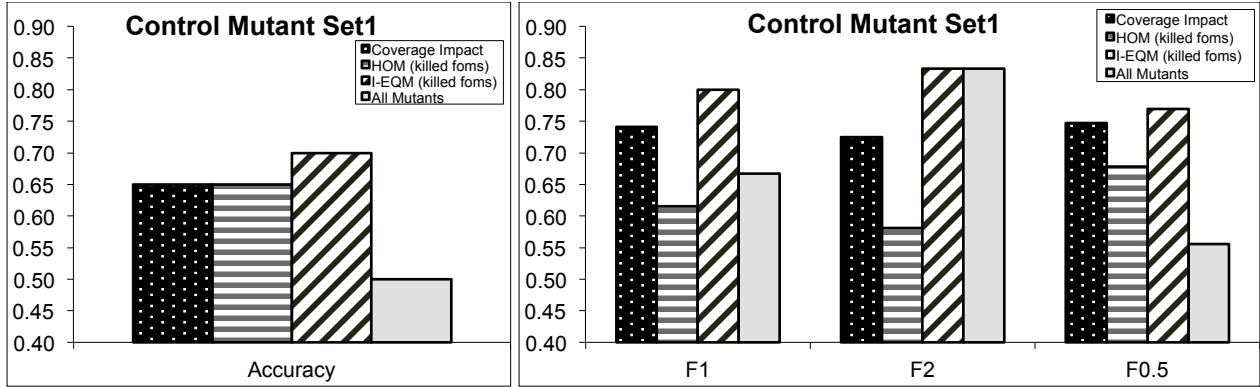


Figure 5. Mutant classifiers comparison for control mutant set 1.

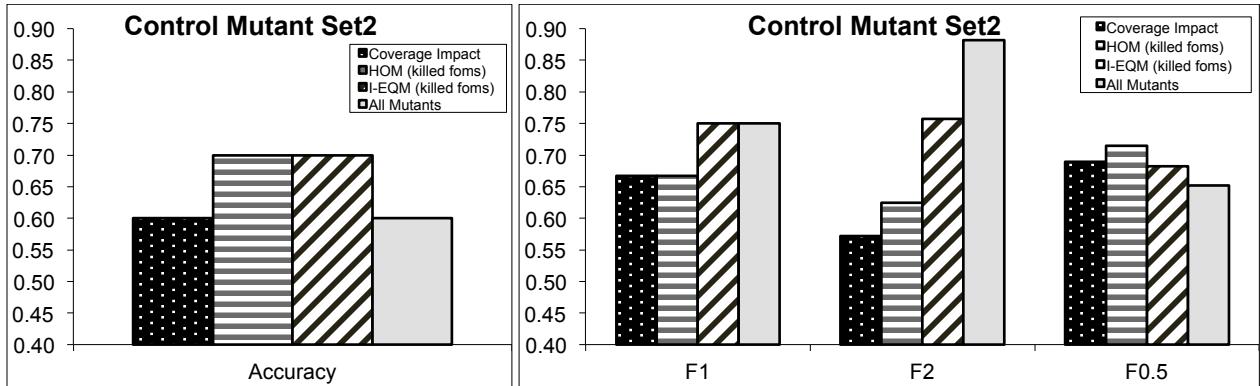


Figure 6. Mutant classifiers comparison for control mutant set 2.

which limits the selection of equivalent mutants, favors 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 winner is the *HOM* classifier.

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 in 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.

D. Stability of the Classifiers

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

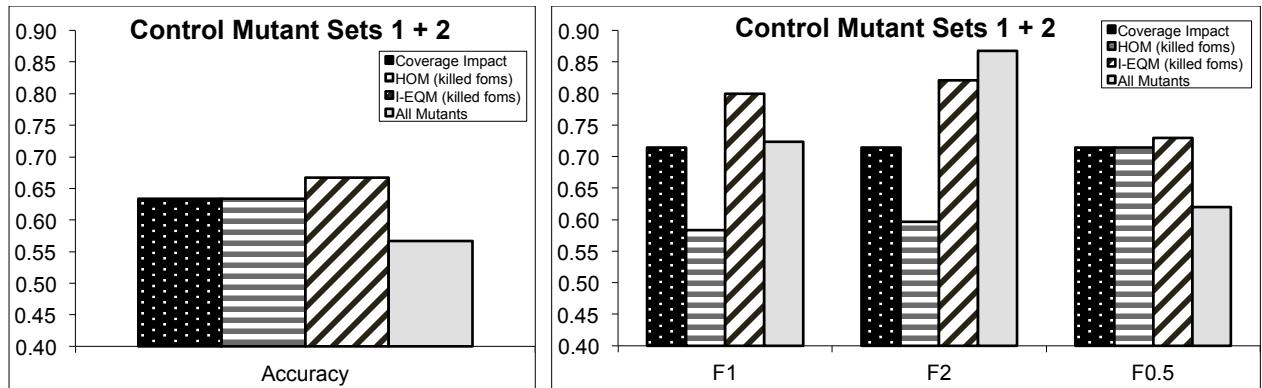


Figure 7. Mutant classifiers comparison for control mutant sets 1 and 2.

18%. With respect to the recall metric, the *Coverage Impact* approach experiences a variation level of 28%, while 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 visualize 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 a small loss of precision and with the highest level of stability for the examined test subjects.

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 utilized, instead of a parametric one, due to the fact that it is based on no distributional assumptions for the considered data observations. A series of two-tailed hypothesis tests were employed 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*

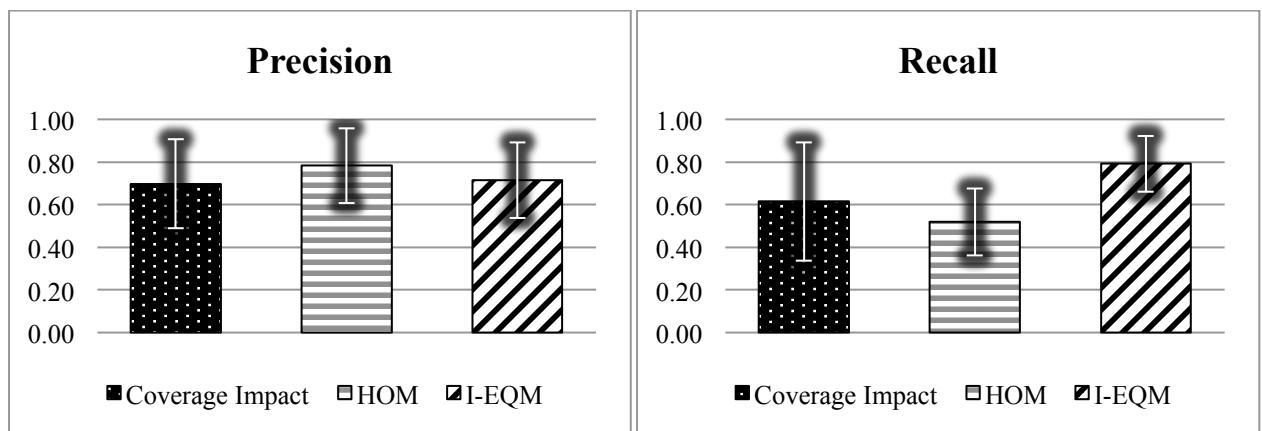


Figure 8. Mean and standard deviation values of the precision and recall metrics for both control mutants sets.

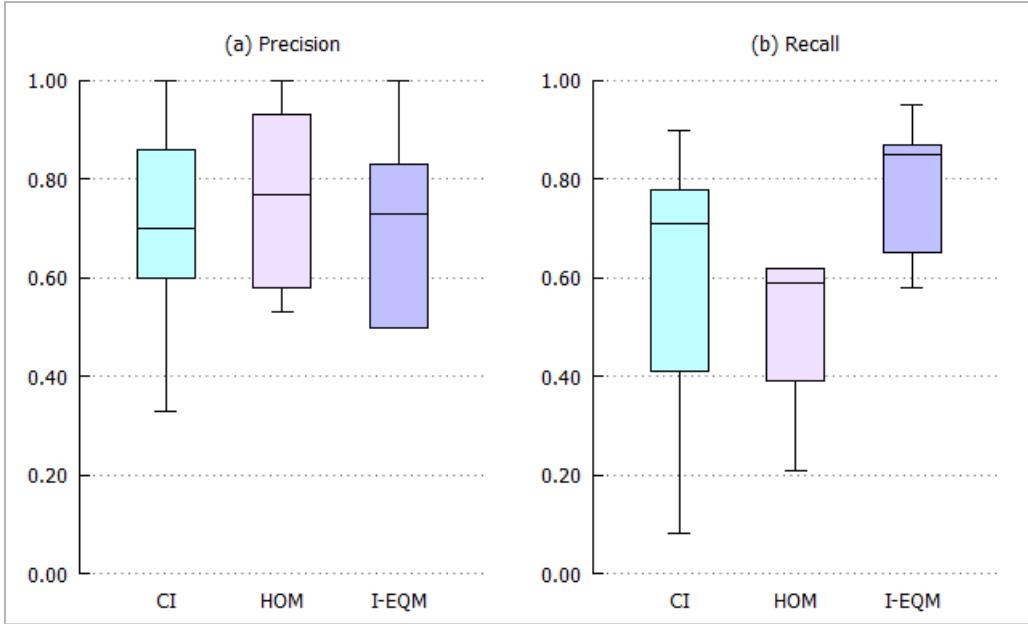


Figure 9. Variation among the precision and recall values of the examined classifiers for both control mutant sets.

Classifier, regarding its precision metric, are the following:

- 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, the *null hypothesis* is rejected only in the case of “*HOM* versus *I-EQM*”, i.e., concerning 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. Regarding 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 sufficient 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. Conclusively, these findings suggest there is statistically insufficient evidence to establish a difference in the performance of the *Coverage Impact* and *I-EQM* classifiers regarding precision, whereas there is a statistically significant difference in their performance regarding recall.

E. 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, i.e., 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

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

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. Examining further the records of Tables VI and VII, it becomes obvious that this approach suffers a decrease of nearly 10% of its recall value compared to the ones of *HOM Classifier (killed foms)* and *HOM Classifier (all foms)* techniques and approximately 20% compared to 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 to the rest of the approaches of the *I-EQM Classifier* and has an enhanced recall of nearly 15% compared to 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 score¹² 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 analyzing 1.17% more equivalent mutants¹³.

VI. MUTANTS WITH HIGHER IMPACT

The behavior 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 classification mutant (CM) set whose output has changed after being combined with the live-unclassified mutant (Section III.D).

¹²These scores were evaluated by counting the mutants killed by the employed tests plus the identified killable mutants of the totally estimated killable ones.

¹³ This number was evaluated by counting the identified equivalent mutants of the totally estimated number of equivalent ones.

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, while 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 realized. These findings suggest that an analogy between mutants with the highest impact and high killability ratios cannot be drawn.

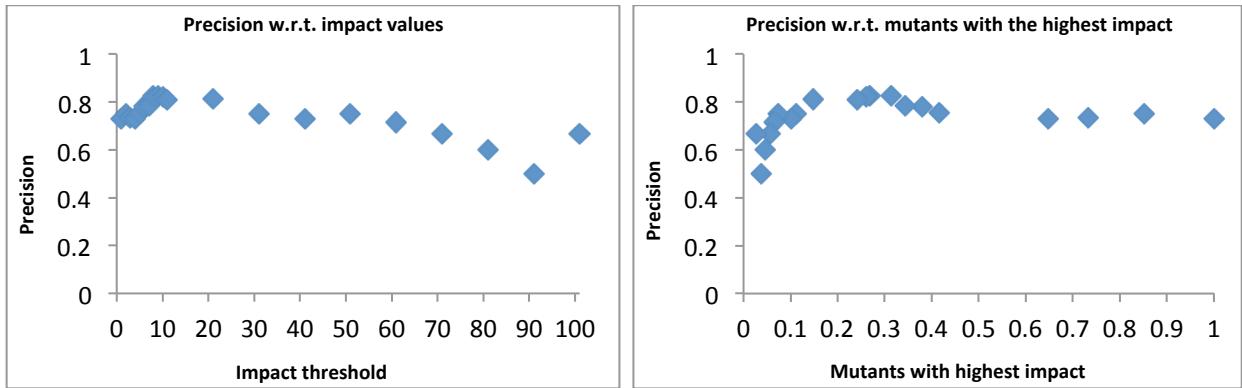


Figure 10. *Coverage Impact Classifier*: Precision w.r.t. Impact values (left) and Mutants with the highest impact (right).

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.

To investigate the differences between the control mutant sets 1 and 2, Figure 12 plots 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, since 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 behavior. 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, since the differences between the two sets are small, it is believed that the above conclusion holds for both studied sets.

VII. 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*. Since the employed subjects are large in size and of different application domains, they are considered as appropriate. Furthermore, many recent empirical studies, such as [17], [19], [20], [22], utilized Javalanche, thus increasing the confidence in its results.

The employed mutant operators constitute another possible issue. Different operator sets, e.g. 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

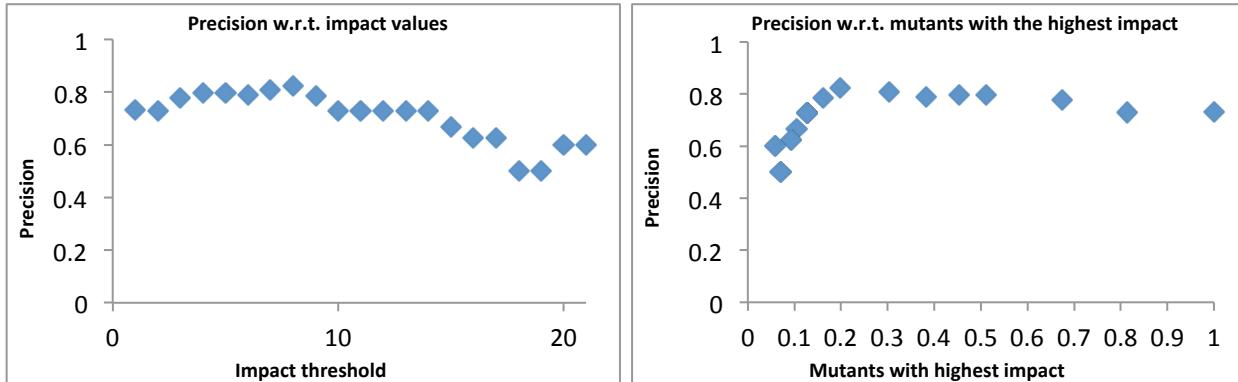


Figure 11. *HOM Classifier*: Precision w.r.t. Impact values (left) and Mutants with the highest impact (right).

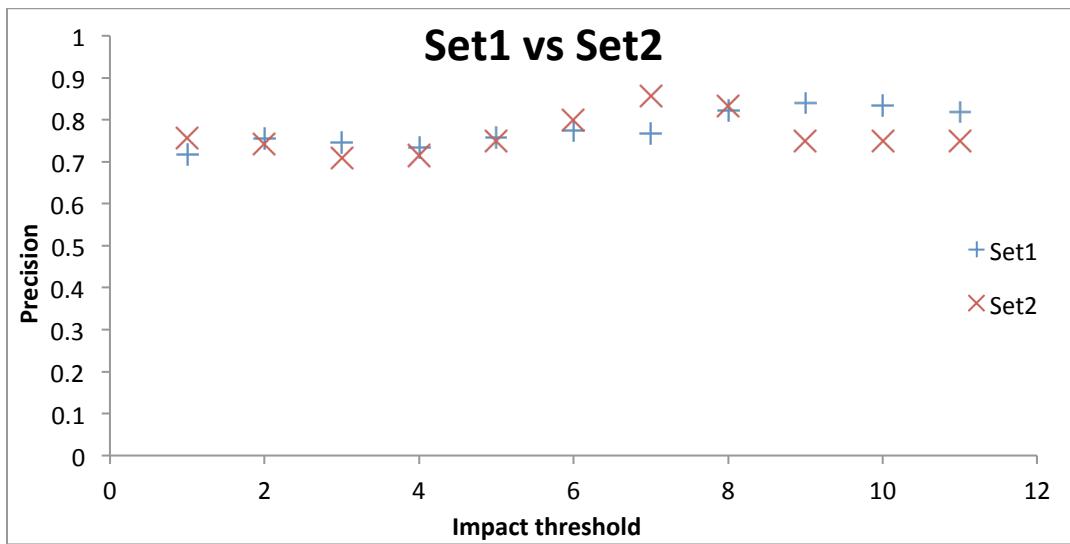


Figure 12. *Coverage Impact Classifier*: Precision w.r.t. Impact values for control mutant set 1 and 2.

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 generalization of its results. The generalization of a study's results is difficult to be achieved due to 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, such as [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 one 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.

VIII. 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 aiming at this issue is due to the work of Adamopoulos et al. [31]. In their attempt, genetic algorithms were employed in order to overcome the difficulties of the large numbers of the introduced mutants and equivalent ones. 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 way 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 Avalanche [11] tool suggested that coverage impact is more efficient and 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, i.e., retrieving most of the killable mutants. Details about the *coverage impact* approach have been given in Section III.

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. Since program optimization produces equivalent program versions, the original and the mutant programs can be optimized or de-optimized 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 on average the 45% of the existing equivalent mutants. Other related approaches make use of program slicing [34] or dependence analysis [35] to aid the tester identify 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 uncategorized 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. To this end, 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.

IX. CONCLUSION

The practical application of mutation testing constitutes the primary aim of the present work. Towards this direction, a dynamic method, *Isolating Equivalent Mutants (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 utilizes 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 ones. **b)** An empirical study based on a manually classified mutant set replicating the *coverage impact* technique. Additionally, **c)** the empirical results reveal that *I-EQM* achieves the correct classification of 81% of the killable mutants with a precision of 71%. Finally, **d)** a manually identified and publicly available set of killable and equivalent mutants is provided. This set amplifies an existing benchmark set [19], [20] with the aim of enabling further studies. In view of this, the first mutant set [19], [20] can be used to develop techniques and the second one can be used to evaluate them.

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 i.e., 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.

REFERENCES

- [1] A. J. Offutt and R. H. Untch, "Mutation 2000: uniting the orthogonal," in *Mutation testing for the new century*, Kluwer Academic Publishers, 2001, pp. 34-44.
- [2] A. P. Mathur and W. E. Wong, "An Empirical Comparison of Data Flow and Mutation-based Test Adequacy Criteria," *Software Testing, Verification and Reliability*, vol. 4, no. 1, pp. 9-31, 1994.

- [3] A. J. Offutt, J. Pan, K. Tewary, and T. Zhang, "An experimental evaluation of data flow and mutation testing," *Softw. Pract. Exper.*, vol. 26, no. 2, pp. 165-176, 1996.
- [4] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.
- [5] N. Li, U. Praphamontripong, and A. J. Offutt, "An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-uses and Prime Path Coverage," in *Proceedings of the 4th International Workshop on Mutation Analysis (MUTATION'09)*, 2009, pp. 220-229.
- [6] J. H. Andrews, L. C. Briand, and Y. Labiche, "Is mutation an appropriate tool for testing experiments?," in *Proceedings of the 27th international conference on Software engineering*, 2005, pp. 402-411.
- [7] J. H. Andrews, L. C. Briand, Y. Labiche, and A. S. Namin, "Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria," *IEEE Trans. Softw. Eng.*, vol. 32, no. 8, pp. 608-624, 2006.
- [8] Y. Jia and M. Harman, "An Analysis and Survey of the Development of Mutation Testing," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 649-678 2011.
- [9] M. Papadakis and N. Malevris, "An Empirical Evaluation of the First and Second Order Mutation Testing Strategies," in *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, 2010, pp. 90-99.
- [10] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf, "An experimental determination of sufficient mutant operators," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 2, pp. 99-118, 1996.
- [11] D. Schuler and A. Zeller, "Javalanche: Efficient Mutation Testing for Java," *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, pp. 297-298, 2009.
- [12] A. J. Offutt and J. Pan, "Automatically Detecting Equivalent Mutants and Infeasible Paths," *Software Testing, Verification and Reliability*, vol. 7, pp. 165-192, 1997.
- [13] T. A. Budd and D. Angluin, "Two notions of correctness and their relation to testing," *Acta Informatica*, vol. 18, no. 1, pp. 31-45, 1982.
- [14] M. Papadakis and N. Malevris, "Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing," *Software Quality Journal*, vol. 19, no. 4, pp. 691-723, 2011.
- [15] M. Papadakis and N. Malevris, "Automatic Mutation Test Case Generation via Dynamic Symbolic Execution," in *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, 2010, pp. 121-130.
- [16] M. Papadakis and N. Malevris, "Mutation based test case generation via a path selection strategy," *Information and Software Technology*, vol. 54, no. 9, pp. 915-932, Sep. 2012.
- [17] G. Fraser and A. Zeller, "Mutation-driven generation of unit tests and oracles," in *Proceedings of the 19th international symposium on Software testing and analysis*, 2010, pp. 147-158.
- [18] B. Baudry, F. Fleurey, J.-M. Jézéquel, and Y. L. Traon, "From genetic to bacteriological algorithms for mutation-based testing: Research Articles," *Softw. Test. Verif. Reliab.*, vol. 15, no. 2, pp. 73-96, 2005.
- [19] D. Schuler and A. Zeller, "(Un-)Covering Equivalent Mutants," in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, 2010, pp. 45-54.
- [20] D. Schuler and A. Zeller, "Covering and Uncovering Equivalent Mutants," *Software Testing, Verification and Reliability*, 2012.
- [21] B. J. M. Grun, D. Schuler, and A. Zeller, "The Impact of Equivalent Mutants," in *Software Testing, Verification and Validation Workshops, 2009. ICSTW '09. International Conference on*, 2009, pp. 192-199.
- [22] D. Schuler, V. Dallmeier, and A. Zeller, "Efficient mutation testing by checking invariant violations," *Proceedings of the eighteenth international symposium on Software testing and analysis*. ACM, Chicago, IL, USA, pp. 69-80, 2009.
- [23] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on Test Data Selection: Help for the Practicing Programmer," *Computer*, vol. 11, no. 4, pp. 34-41, 1978.
- [24] A. J. Offutt, "Investigations of the software testing coupling effect," *ACM Trans. Softw. Eng. Methodol.*, vol. 1, no. 1, pp. 5-20, 1992.
- [25] Y. Jia and M. Harman, "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, pp. 249-258, 2008.
- [26] Y. Jia and M. Harman, "Higher Order Mutation Testing," *Inf. Softw. Technol.*, vol. 51, no. 10, pp. 1379-1393, 2009.
- [27] M. Polo, M. Piattini, and I. García-Rodríguez, "Decreasing the cost of mutation testing with second-order mutants," *Software Testing, Verification and Reliability*, vol. 19, no. 2, pp. 111-131, 2009.
- [28] M. Kintis, M. Papadakis, and N. Malevris, "Evaluating Mutation Testing Alternatives: A Collateral Experiment," in *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*, 2010, pp. 300-309.
- [29] M. Kintis, M. Papadakis, and N. Malevris, "Isolating First Order Equivalent Mutants via Second Order Mutation," *Software Testing, Verification, and Validation, 2012 International Conference on*, vol. 0, pp. 701-710, 2012.
- [30] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

- [31] K. Adamopoulos, M. Harman, and R. M. Hierons, "How to Overcome the Equivalent Mutant Problem and Achieve Tailored Selective Mutation Using Co-evolution," *GECCO* (2), vol. 3103. Springer, pp. 1338-1349, 2004.
- [32] D. Baldwin and F. G. Sayward, "Heuristics for Determining Equivalence of Program Mutations," Yale University, New Haven, Connecticut, 1979.
- [33] A. J. Offutt and W. M. Craft, "Using Compiler Optimization Techniques to Detect Equivalent Mutants," *Software Testing, Verification and Reliability*, vol. 4, no. 3, pp. 131-154, 1994.
- [34] R. M. Hierons, M. Harman, and S. Danicic, "Using Program Slicing to Assist in the Detection of Equivalent Mutants," *Software Testing, Verification and Reliability*, vol. 9, no. 4, pp. 233-262, 1999.
- [35] M. Harman, R. M. Hierons, and S. Danicic, "The Relationship Between Program Dependence and Mutation Analysis," *Proceedings of the 1st Workshop on Mutation Analysis (MUTATION'00)*. San Jose, California, pp. 5-13, 2001.
- [36] W. B. Langdon, M. Harman, and Y. Jia, "Efficient multi-objective higher order mutation testing with genetic programming," *J. Syst. Softw.*, vol. 83, no. 12, pp. 2416-2430, 2010.
- [37] W. B. Langdon, M. Harman, and Y. Jia, "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, pp. 21-29, 2009.
- [38] J. Yue and M. Harman, "MILU: A Customizable, Runtime-Optimized Higher Order Mutation Testing Tool for the Full C Language," in *Practice and Research Techniques, 2008. TAIC PART '08. Testing: Academic & Industrial Conference*, 2008, pp. 94-98.
- [39] M. Harman, Y. Jia, and W. B. Langdon, "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, pp. 212-222, 2011.
- [40] M. Harman, Y. Jia, and W. B. Langdon, "A Manifesto for Higher Order Mutation Testing," *Proceedings of the 5th International Workshop on Mutation Analysis (MUTATION'10)*. Paris, France, pp. 80-89, 2010.