Associating Code Clones with Association Rules for Change Impact Analysis

Manishankar Mondal Banani Roy Chanchal K. Roy Kevin A. Schneider
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
{mshankar.mondal, banani.roy, chanchal.roy, kevin.schneider}@usask.ca

***Abstract*—When a programmer makes changes to a target program entity (files, classes, methods), it is important to identify which other entities might also get impacted. These entities constitute the impact set for the target entity. Association rules have been widely used for discovering the impact sets. However, such rules only depend on the previous co-change history of the program entities ignoring the fact that similar entities might often need to be updated together consistently even if they did not co-change before. Considering this fact, we investigate whether cloning relationships among program entities can be associated with association rules to help us better identify the impact sets.**

In our research, we particularly investigate whether the impact set detection capability of a clone detector can be utilized to enhance the capability of the state-of-the-art association rule mining technique, Tarmaq, in discovering impact sets. We use the well known clone detector called NiCad in our investigation and consider both regular and micro-clones. Our evolutionary analysis on thousands of commit operations of eight diverse subject systems reveals that consideration of code clones can enhance the impact set detection accuracy of Tarmaq with a significantly higher precision and recall. Micro-clones of 3LOC and 4LOC and regular code clones of 5LOC to 20LOC contribute the most towards enhancing the detection accuracy.

***Index Terms*—Change Impact Analysis, Evolutionary Coupling, Code Clones**

I. Introduction

Changes are inevitable to a software system during its maintenance and evolution. However, making changes to a software system's code-base is often critical because changes to a particular program entity might have impacts on some other related entities. Identification of these related entities that might get impacted is an important software engineering task which is known as change impact analysis in the literature [28]. Making changes to a program entity without properly analyzing the impact of the changes might introduce bugs and inconsistencies in the code-base. Evolutionary coupling [45], [47] has been widely used for change impact analysis.

If two or more program entities changed together frequently in the past, then we realize a coupling among these entities. This coupling is known as evolutionary coupling [47]. If a group of program entities exhibited evolutionary coupling during the past evolution, then it is expected that a change in one entity in future will require corresponding changes to the other entities in the group. These other entities constitute the impact set according to evolutionary coupling. Evolutionary coupling has been realized using association rules [47]. We will discuss association rules in Section II.

The capability of evolutionary coupling (i.e., association rules) in identifying the impact set is limited because it only depends on the past co-change (changing together) history of program entities. For example, if a programmer attempts to make changes to an entity E_1, then another entity E_2 will not be considered in the impact set of E_1 if E_1 and E_2 did not change together in the past. However, E_2 might also get impacted because of the changes to E_1 even if these two entities did not co-change before.

Focusing on the above drawback of evolutionary coupling in change impact analysis, in this paper we investigate whether code clones residing in the code-base of a software system can help us in an improved detection of the impact sets. Code clones are identical or nearly similar code fragments in a software system's code-base [7]. These are mainly created because of the frequent copy/paste activities of the programmers during development. Two code fragments that are similar to each other make a clone-pair. A group of similar code fragments forms a clone class (i.e., clone group). Existing studies [35], [36] show that code clones from the same clone class have a tendency of getting changed together consistently during evolution. We investigate whether we can utilize this tendency in association with evolutionary coupling for better detection of impact sets for program entities.

Our study involves the state of the art technique called *Tarmaq* [45] which can detect impact sets of program entities using evolutionary coupling. We combine clone analysis with Tarmaq and investigate whether the combined technique can detect impact sets with better accuracy compared to the original one. For detecting and analyzing code clones, we use the NiCad clone detection tool [23] which is a promising one among the modern clone detectors. We detect and consider both regular and micro-clones [30], [33], [44] in our research. While the minimum size of the regular code clones is 5 LOC as was suggested by Wang et al. [46], micro-clones can be of at most 4 LOC according to the literature [33]. Existing studies show that micro-clones have a tendency of getting changed together consistently during evolution. We conduct our research by considering method level granularity. Thus, the program entities in our research are methods. While most of the existing studies on evolutionary coupling considered file level granularity, we consider a finer granularity (i.e., method level granularity) because it can help us pin point which methods in a file should be included in the impact set. To the best of our knowledge, ours is the first study to investigate the

978-1-7281-5143-4/20/$31.00 © 2020 IEEE

TABLE I
INVESTIGATED RESEARCH QUESTIONS

Serial	Research question
RQ 1	Does a combination of clone analysis and evolutionary coupling detect impact sets with better accuracy compared to evolutionary coupling alone?
RQ 2	Are regular or micro-clones more beneficial in retrieving impact sets?
RQ 3	What sizes of micro-clones are more beneficial to change impact analysis?
RQ 4	Which sizes of regular clones are more beneficial to change impact analysis?
RQ 5	Does a combination of clone analysis and evolutionary coupling detect impact sets more often compared to evolutionary coupling alone during system evolution?

role of clone analysis in association with evolutionary coupling towards an improved detection of impact sets considering method granularity.

We perform our investigation on thousands of commits of eight diverse subject systems written in three different programming languages (Java, C, C#) and answer the research questions listed in Table I by analyzing our experiment results. We have the following findings:

- Clone analysis can significantly improve (according to our statistical significance tests) the impact set detection accuracy (precision, recall) of the state of the art association rule mining based technique called *Tarmaq*.
- Micro-clones of 3LOC and 4LOC are more beneficial than the other sizes of micro-clones towards improving the impact set detection accuracy of Tarmaq.
- Regular clones within the range of 5 to 20LOC contribute the most towards improved detection of impact sets.

Our findings are important for devising mechanisms for predicting impact sets considering method level granularity. If we use the NiCad clone detector with Tarmaq for the purpose of detecting impact sets, code clones within the range of 3 to 20LOC are the most beneficial ones. Code clones beyond this range can be disregarded because such clones have a very low probability of providing true positive co-change candidates for a target program entity (method in our research).

The rest of the paper is organized as follows. Section II discusses the background topics, Section III describes the existing studies related to our research, Section IV presents the way we combine clone analysis and evolutionary coupling, Section V describes the experiment setup and steps, Section VI answers the research questions by discussing and analyzing our experiment results, Section VII mentions some possible threats to validity, and finally, Section VIII concludes the paper by mentioning possible future work.

II. BACKGROUND

A. Evolutionary Coupling

Evolutionary coupling among program entities is a well investigated phenomenon in software engineering research and practice. This coupling can be realized using *association rules* [42], [47] with two related measures: *Support* and *Confidence*.

Association Rule. An association rule [42] is formally defined as an expression of the form $X => Y$. Here, X is known as the antecedent and Y is the consequent. Each of X and Y is a set of one or more program entities. In the context of software engineering, such an association rule implies that if X gets changed in a particular commit operation, Y also has the tendency of getting changed in that commit operation.

Support and confidence of an association rule. According to Zimmermann et al. [47], *support is the number of commits in which an entity or a group of entities changed together*. Let X is the set of one or more program entities. C_X is the set of commits such that all the program entities in X changed together in each of these commits. Then, support of X can be calculated using the following equation.

$$Support(X) = |C_X| \tag{1}$$

Now, the support of the association rule $X => Y$ where each of X and Y is a set of one or more program entities can be calculated using the following equation.

$$Support(X => Y) = Support(X \cup Y) \tag{2}$$

Here, $X \cup Y$ is the union of the sets X and Y. *Confidence of an association rule, $X => Y$, determines the conditional probability that Y will change in a commit operation provided that X changed in that commit operation*. We determine the confidence of the association rule, $X => Y$, using Eq. 3.

$$Confidence(X => Y) = \frac{Support(X => Y)}{Support(X)} \tag{3}$$

Let us now consider an example of two program entities E_1 and E_2. These entities can be files, classes, or methods. If E_1 and E_2 have ever changed together (co-changed), we can assume two association rules, $E_1 => E_2$ and $E_2 => E_1$ from these. Suppose, E_1 was changed in four commits: 2, 5, 6, and 10 and E_2 was changed in six commits: 4, 6, 7, 8, 10, and 13. Thus, according to the definition, *Support(E_1)* = 4 and *Support(E_2)* = 6. *Support(E_1, E_2)* = 2, because E_1 and E_2 changed together (co-changed) in two commits: 6 and 10. Again, $Support(E_1 => E_2) = Support(E_2 => E_1) = Support(E_1, E_2) = 2$. Now, *confidence ($E_1 => E_2$) = support($E_1$, E_2) / support(E_1) = 2 / 4 = 0.5* and *confidence($E_2 => E_1$) = 2 / 6 = 0.33*.

In our experiment, an association rule, $X => Y$, is such that each of the sets X and Y consists of a single method.

B. Regular Code Clones

Our research involves detection and analysis of regular code clones of all three major clone-types: Type 1, Type 2, and Type 3. We define these clone-types in the following way according to the literature [4], [7].

If two or more code fragments in a code-base are exactly the same disregarding their comments and indentations, these code fragments are identical clones or **Type 1 clones** of one another. Syntactically similar code fragments residing in a software system's code-base are known as **Type 2 clones**. Type

2 clones are generally created from Type 1 clones because of renaming identifiers and/or changing data types. Finally, **Type 3 clones**, also known as gapped clones, generally get created from Type 1 or Type 2 clones because of additions, deletions, or modifications of lines in these clones. In our research, we detect regular code clones of at least 5 LOC which is the best minimum threshold for detecting regular code clones according to Wang et al. [46].

C. Micro-Clones

According to the literature [30], [33], [44], micro-clones are smaller than the minimum size threshold of the regular code clones. As we detect regular clones of at least 5 LOC, micro-clones can be of at most 4 LOC in our research. The minimum size of a micro-clone fragment can be 1LOC. Mondal et al. [33] showed that micro-clones have a tendency of co-changing (changing together) consistently during evolution.

D. Method Genealogy

During the evolution of a software system, a particular method might be created in a particular revision and the method might remain alive in a number of consecutive revisions. Each of these revisions contains a snap-shot of the method. The genealogy of the method consists of the snap-shots of it from all those revisions where it was alive. Thus, the genealogy of a particular method can help us analyze how it was changed over the evolution.

III. Related Work

A. Change Impact Analysis Using Evolutionary Coupling

Change impact analysis is a well investigated area of research in the field of software engineering. A great many studies [10], [15], [17], [22], [28] have been done on change impact analysis resulting a number of impact analysis techniques and tools. Evolutionary coupling has been investigated by a number of studies [13], [18], [19], [31] for change impact analysis. Existing studies [16], [20] show that evolutionary coupling, also known as logical coupling, can discover those coupling links that are missed by other impact analysis techniques.

Kagdi et al. [20] performed change impact analysis (CIA) by using evolutionary coupling in combination with conceptual coupling. They found that a combination of these two couplings can perform better change impact analysis than each individual. Our study is different because we investigate whether clone analysis can be combined with evolutionary coupling for improved detection of impact sets. Moreover, conceptual coupling which was used by Kagdi et al. [20] is only applicable to object oriented systems [11]. Our research involves non-object-oriented systems (such as Ctags, Camellia, and BRL-CAD as mentioned in Table II) too.

Rolfsness et al. [45] proposed a technique called *Tarmaq* for better detection of evolutionary coupling. They showed that *Tarmaq* performs better than ROSE [47] in detecting evolutionary coupling. In our research, we combine clone analysis with Tarmaq and find that the combined technique can perform significantly better than Tarmaq in detecting impact sets considering method granularity.

Pugh et al. [52] investigated whether it is possible to achieve the same level of accuracy like Tarmaq through analyzing a dynamically selected smaller number of commit transactions than Tarmaq. Islam et al. [29] introduced the concept of *Transitive Association Rules*. They first detected the regular association rules and then applied transitivity on these rules to make new rules for better detection of evolutionary coupling. Our study is different from these studies. We investigate whether clone analysis can be beneficial towards detecting impact sets during evolution. Our investigation involving the state of the art technique called Tarmaq implies that clone analysis combined with Tarmaq can better detect impact sets considering method level granularity.

A number of existing studies [1], [13], [14], [18], [38], [43] have investigated improving the detection accuracy of evolutionary coupling by combining association rule mining with the Granger causality test [18], macro co-change and dephase macro co-change [14], and interactions [13]. Some preliminary studies have also investigated several new measures such as *significance* [38], pattern age and pattern distance [1], and change correspondence [37] for detecting evolutionary coupling with higher accuracy.

Our study is different from all these existing studies because we investigate whether cloning relationship among code fragments can be utilized for change impact analysis. In particular, we investigate whether clone analysis in association with evolutionary coupling can better detect impact sets compared to evolutionary coupling alone. To the best of our knowledge, our study is the first one to investigate this matter. From our analysis on the state-of-the-art evolutionary coupling detection technique, *Tarmaq*, we find that a combination of clone analysis and evolutionary coupling performs significantly better than just evolutionary coupling in terms of precision and recall.

B. Studies on Code Clones

Code clones have been heavily investigated by a great many studies [4]–[8], [12], [21], [26], [27], [32], [33]. Existing studies have controversial findings [12], [21], [26] regarding the impacts of code clones on software maintenance and evolution. While some of the studies [25], [26], [40] show that code clones have positive impacts on software maintenance, a number of studies [2], [3], [12], [34], [39] have shown evidences of negative impacts of code clones. In our study we investigate whether we can analyze code clones in association with evolutionary coupling for better detection of impact sets. A number of studies [35], [36] show that code clones from the same clone class have a tendency of changing together consistently. We utilize this tendency with evolutionary coupling for better detection of impact sets. Our research shows that clone analysis can significantly improve the detection accuracy of impact sets of the state-of-the-art evolutionary coupling based technique called *Tarmaq*.

As clone analysis helps us achieve promising results in change impact analysis when combined with evolutionary

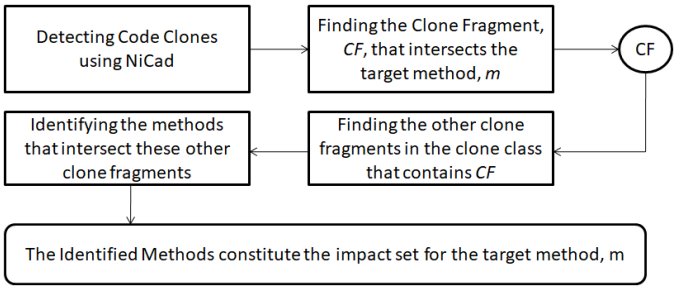

Fig. 1. Detecting the impact set for a target method through applying NiCad

coupling, we expect that it will also provide good results in association with other techniques and measures such as Macocha [14], significance [38], and change correspondence [37]. We plan to investigate this in future. Also, most of the existing studies on evolutionary coupling investigated file level coupling. Our study is more fine grained (method granularity), and thus, we can help programmers pin point which fine grained entities (i.e., methods) instead of files should be considered in the impact set.

IV. Impact Set Detection

Let us assume that a software system is being developed under version control system (e.g., SVN, or GitHub). The latest revision (i.e., the working revision) of the subject system is r. Let us also assume that a programmer is now going to make changes to a target method m residing in revision r. In this situation, we determine the impact set (the other methods in revision r that might also need to be changed with m for ensuring consistency of the code-base) for m using the following mechanisms.

A. Detecting Impact Sets Using Tarmaq

Rolfsness et al. [45] introduced Tarmaq for detecting impact sets considering file level granularity. Thus, Tarmaq can predict/suggest which other files might get impacted when a programmer attempts to make changes to a particular target file. For making this file level suggestions, Tarmaq analyzes the past evolution history of the software system, identifies which files co-changed in which commit operations, and finally, determines file level association rules for a particular query using the algorithm proposed by Rolfsness et al. [45]. In our research, we use Tarmaq for predicting impact sets considering method level granularity. For this purpose, we first detect which methods co-changed in which commit operations, and then determine method level association rules using the same algorithm [45] for a particular query. For extracting method co-change information, we detect method genealogies by applying the technique proposed by Lozano and Wermelinger [2]. In the case of method level granularity, the query consists of methods. In the first paragraph of Section IV we mentioned that we are going to find the impact set for a target method m which a programmer attempted to change. In this situation, the query for Tarmaq consists of m.

TABLE II
The subject systems that we investigate

Systems	Lang.	Domains	LLR	SRev	LRev
Ctags	C	Code Definition Generator	33,270	1	774
Camellia	C	Multimedia	85,015	1	207
BRL-CAD	C	Solid modeling system	39,309	1	735
Carol	Java	Game	25,091	1	1700
Freecol	Java	Game	91,626	1000	1950
DNSJava	Java	DNS Protocol	23,373	1	1635
Greenshot	C#	Multimedia	37,628	1	999
MonoOSC	C#	Formats and protocols	14,883	1	355

LLR = LOC in the Last Revision
SRev = Starting Revision LRev = Last Revision

B. Detecting Impact Sets Using Tarmaq and NiCad

Section IV-A discusses the way how we detect impact set for the target method m by using it as the query to the Tarmaq algorithm [45]. After determining the impact set using Tarmaq, we apply NiCad for finding which other methods have cloning relationships with m using the procedure depicted in Fig. 1.

As demonstrated in Fig. 1, we apply NiCad to find all the clone classes from the code-base. We check each of the clone fragments from each of the clone classes and determine which fragment intersects with m. If we get such a fragment, it implies that the programmer is going to make changes inside a clone fragment. Let us denote this clone fragment which the programmer has attempted to change by CF. As code clones have a tendency of changing together consistently, the other clone fragments in the clone class that contains CF might also need to changed consistently with CF. Considering this fact we determine the other clone fragments in the clone class that contains CF. The methods that intersect these other clone fragments are considered as the impact set for method m. Thus, we get the following two impact sets for m:

(1) One set was obtained by analyzing the past evolutionary history by applying Tarmaq.

(2) The other set was obtained by NiCad through detecting code clones from the code-base.

We make a union of these two sets to find the final impact set for the target method m.

V. Experiment Setup and Steps

We conduct our experiment on eight software systems listed in Table II. We download these systems from an on-line SVN repository called SourceForge.net [41]. Table II shows the subject systems along with their application domains, starting revisions, ending revisions, implementation languages, and sizes (LOC). For most of the subject systems (i.e., except Freecol) the starting revision is 1. However, for Freecol, the starting revision is 1000. The former revisions of Freecol are not available in the on-line repository. For each subject system we perform the following preliminary steps.

- Downloading all the revisions (as indicated in Table II) of the subject system from their repositories.
- Detecting methods from each of the revisions using the tool called CTAGS [9].

- Detecting three types of code clones from each revision using the clone detection tool called NiCad [23].
- Detecting changes between every two consecutive revisions using UNIX *diff* operation.
- Mapping changes to the already detected methods in each revision by using the starting and ending line numbers of the methods and changes.
- Detecting method genealogies by considering the methods from all the revisions following the technique that was proposed by Lozano and Wermelinger [2].

Detection of regular and micro clones using NiCad. As we mentioned before, we detect regular code clones from the open source subject systems using the clone detection tool, NiCad [23]. An existing research [24] shows that NiCad is a promising choice among the modern clone detectors because it shows high precision and recall in detecting the major three types (Type 1, Type 2, and Type 3) of regular clones. As was done by Svajlenko and Roy [24], we apply NiCad to detect block level code clones of at least 5 lines and at most 500 lines considering a dissimilarity threshold of 30% with blind-renaming of identifiers. We also detect micro-clones of 1 to 4LOC from our subject systems using NiCad.

VI. Experimental Results and Analysis

In this section, we answer the research questions listed in Table I by analyzing our experiment results and determine whether and how consideration of code clones with evolutionary coupling can result in better detection of impact sets.

A. Answering the First Research Question (RQ 1)

RQ 1: *Does a combination of clone analysis and evolutionary coupling detect impact sets with better accuracy compared to evolutionary coupling alone?*

Answering RQ 1 is the primary goal of our research. Tarmaq is the state of the art technique that can suggest impact sets on the basis of evolutionary coupling. If it is observed that consideration of clone analysis with this technique can help it better detect impact sets, this will establish clone analysis to be beneficial for change impact analysis. We perform our investigation in the following way.

Investigation procedure. Section IV describes two impact set detection techniques. We use the following abbreviations for these techniques for the ease of our discussion.

(1) Tarmaq: This was introduced by Rolfsness et al. [45].

(2) Tarmaq-NiCad: It represents the technique that combines clone analysis with Tarmaq. We perform clone analysis by detecting code clones using NiCad [23].

We determine the accuracy of each of these techniques in predicting impact sets in the following way.

Determining the accuracy (precision and recall) of the impact set detection techniques. Let us assume that the evolutionary history of a subject system consists of C commit operations. We sequentially examine each of these commit operations from the very beginning one. Let us consider a particular commit c_i which was applied on revision r_i. When examining the commit operation c_i, we determine which methods changed together in this commit. Let us assume that the methods m_1, m_2, m_3, and m_4 changed together in this commit operation. Now, if we consider m_1, then we can say that the other three methods (i.e., m_2, m_3, and m_4) constitute the actual impact set for m_1. Our goal is to determine how accurately we can determine this actual impact set (m_2, m_3, and m_4) for m_1 by applying the four impact set detection techniques described in Section IV.

Let us assume that A is the actual impact set for method m_1 that we obtained by analyzing commit c_i. The impact set that we obtained by applying a particular impact set detection technique is S. Then, the precision and recall of that particular technique in retrieving the impact set can be calculated using the following two equations.

$$Precision = \frac{|A \cap S| \times 100}{|S|} \tag{4}$$

$$Recall = \frac{|A \cap S| \times 100}{|A|} \tag{5}$$

By considering each of the methods that was changed in each of the commit operations of a subject system, we determine the precision and recall of each of the two techniques in detecting impact set. We finally determine the average precision and recall of each technique by considering all the methods that were changed in all the commit operations. Table III records the total number of method change cases (i.e., the row with SL no. = 1) that we investigated from each of our subject systems. We consider the method change cases from each commit operation where at least two or more methods got changed. The other commits cannot help us in analyzing the accuracy of the impact set detection techniques.

Comparing the accuracy of Tarmaq and Tarmaq-NiCad. For determining whether clone analysis can improve the impact set prediction accuracy of Tarmaq, we compare the two techniques Tarmaq and Tarmaq-NiCad with respect to their precision and recall values.

Comparison regarding recall. Fig. 2 makes a comparison of the average recalls of the two techniques (Tarmaq, and Tarmaq-NiCad) in detecting impact sets. We can see that the recall of Tarmaq-NiCad is always (i.e., for each subject system) higher than that of Tarmaq. The reason behind the higher recalls of Tarmaq-NiCad is that it retrieves more true positive methods (i.e., correct suggestions) to include in the impact set compared to Tarmaq. As Tarmaq-NiCad performs clone analysis in association with evolutionary coupling, it can retrieve a higher number of true positives. Tarmaq only considers evolutionary coupling for finding impact sets. We wanted to know whether the recalls provided by Tarmaq-NiCad are significantly better than those provided by Tarmaq. We conduct statistical significance test for this purpose. Our next paragraph elaborates on our test.

Statistical significance tests regarding comparing recalls. We wanted to analyze whether the recalls of Tarmaq-NiCad are significantly higher than those of Tarmaq. For this purpose, we conduct Wilcoxon Signed Rank test [48], [49] considering the

TABLE III
STATISTICS REGARDING THE NUMBER OF CASES WHERE TARMAQ OR TARMAQ-NICAD PROVIDE TRUE POSITIVES

Measure	Ctags	Camellia	Brlcad	Carol	Freecol	DNSJava	Greenshot	Mono.
Total number of method change cases	728	467	1257	1717	3026	3501	2057	1065
No. of cases where Tarmaq provides true positives	262	258	611	376	1148	1750	867	606
No. of cases where Tarmaq-NiCad provides true positives	408	308	845	1045	1824	2689	1250	727

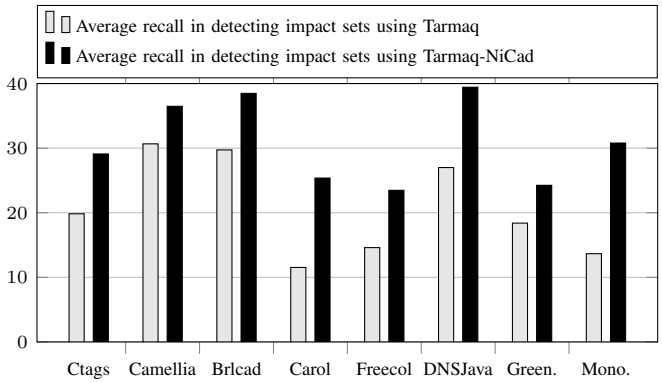

Fig. 2. Comparing the recalls of the techniques Tarmaq and Tarmaq-NiCad for detecting impact sets

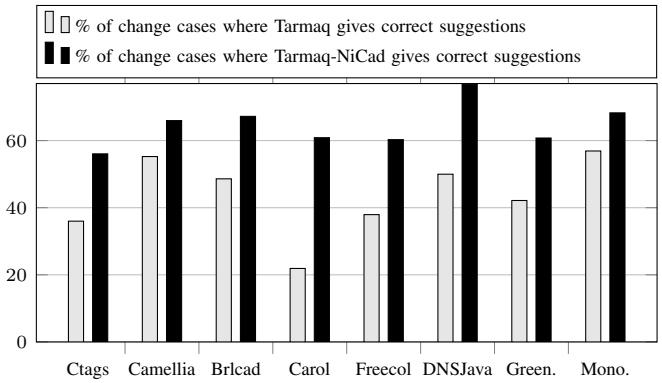

Fig. 4. Comparing the percentages of change cases where Tarmaq and Tarmaq-NiCad can provide impact sets with correct suggestions

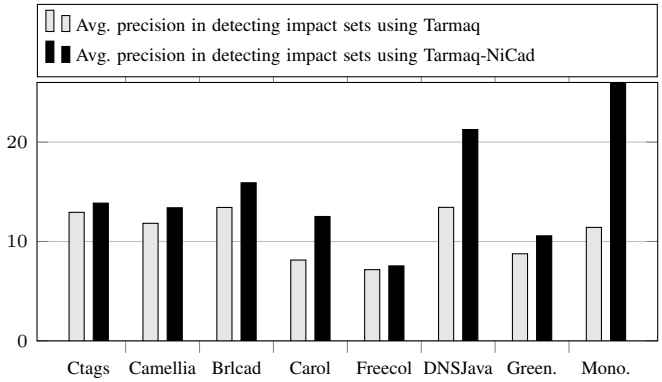

Fig. 3. Comparing the precisions of the techniques Tarmaq and Tarmaq-NiCad for detecting impact sets

recall values plotted in Fig. 2. Wilcoxon Signed Rank test is a non-parametric test which is applicable to paired samples [48]. As this test is non-parametric, it does not require the samples to be normally distributed. We apply this test in our experiment because the recall values that we want to compare are paired. For each subject system, we get two recall values: one from Tarmaq-NiCad and the other from Tarmaq. We should also note that Wilcoxon Signed Rank test can be applied to both small and large data sets [48]. We conduct this test considering a significance level of 5%. According to the test result for the two-tailed test case, the recalls provided by Tarmaq-NiCad are significantly different than those of Tarmaq with a p-value of 0.01 which is less than 0.05. As the recall of Tarmaq-NiCad is always higher than that of Tarmaq, we realize that the recalls of Tarmaq-NiCad are significantly better (higher) than the recalls of Tarmaq. The Cohen's d effect size [51] for the test is 0.630 which is larger than the medium effect size [50].

Comparison regarding precision. Fig. 3 compares the precisions of the two techniques: Tarmaq and Tarmaq-NiCad. We see that the comparison scenario regarding precision is similar to the comparison scenario regarding recalls (Fig. 2). For each of the subject systems, the average precision of Tarmaq-NiCad is higher than the average precision of Tarmaq. The reason behind the higher precision of Tarmaq-NiCad is that through clone analysis it adds true positives to the impact set retrieved using evolutionary coupling.

Statistical significance tests regarding comparing precisions. We again perform Wilcoxon Signed Rank test [48], [49] to determine whether the precisions of Tarmaq-NiCad for different subject systems are significantly higher than the precisions of Tarmaq. As we did previously, we conducted Wilcoxon Signed Rank test considering a significance level of 5%. Our test result for two tailed test case implies that the precisions of Tarmaq-NiCad are significantly different than the precisions of Tarmaq with a p-value of 0.01 which is less than 0.05. As Tarmaq-NiCad's precision is always higher, we can realize that the combined technique (Tarmaq-NiCad) has a significantly higher precision in detecting impact sets for methods compared to Tarmaq. The Cohen's d effect size [50], [51] for the test is 0.630.

Comparison regarding the number of cases where the two candidate techniques (Tarmaq and Tarmaq-NiCad) can provide correct suggestions. We also wanted to find the number of method changes cases where Tarmaq-NiCad or Tarmaq provides impact sets with true positives (i.e., correct suggestions). Table III shows these numbers. We see that the number regarding Tarmaq-NiCad is always higher than the number regarding Tarmaq. Fig. 4 shows the percentages of these two numbers with respect to all change cases for each subject system. Using Wilcoxon Signed Rank Tests [48], [49] as we did previously we find that the percentages regarding Tarmaq-NiCad are significantly higher than those of Tarmaq with a p-value of 0.01 (< 0.05) and a Cohen's d effect size [50], [51] of 0.630.

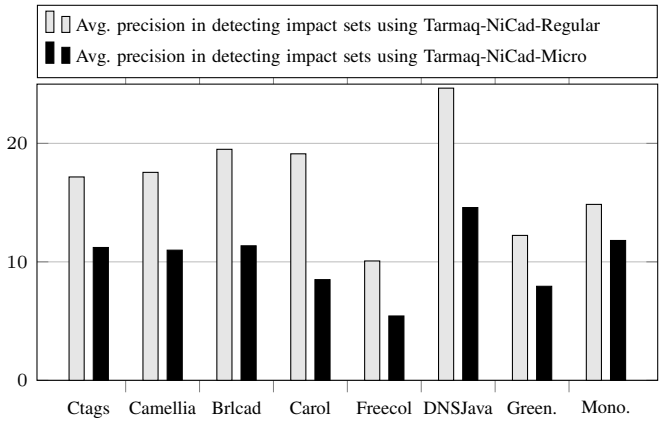

Fig. 5. Comparing the precisions of the techniques Tarmaq-NiCad-Regular and Tarmaq-NiCad-Micro for detecting impact sets

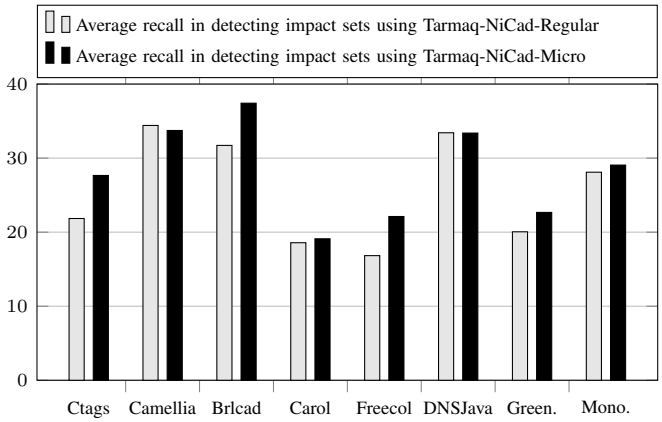

Fig. 6. Comparing the recalls of the techniques Tarmaq-NiCad-Regular and Tarmaq-NiCad-Micro for detecting impact sets

Findings from our analysis. According to our analysis of experiment results, clone analysis can significantly improve the impact set prediction accuracy of the state-of-the-art evolutionary coupling based technique. In other words, a combination of clone analysis and evolutionary coupling can perform significantly better in detecting impact sets compared to evolutionary coupling alone.

We should note that according to the existing studies [20], [47], the precision and recall in detecting impact sets using evolutionary coupling considering method level granularity are generally very low. Kagdi et al. [20] achieved at best 28% recall and 9% precision. Our experiment results depicted in Fig. 2 and 3 also indicate this. In our research, the combined technique (Tarmaq-NiCad) shows the highest recall of 39.43% for our subject system, DNSJava, as shown in Fig. 2. From Fig. 3 we realize that the combined technique provides the highest precision for MonoOSC. This precision is around 26.28%. Although the precision and recall values are small, our findings imply that clone analysis should be considered important when analyzing change impacts, because it can significantly improve the accuracy of detecting impact sets.

B. Answering the Second Research Question (RQ 2)

RQ 2: *Are regular or micro-clones more beneficial in retrieving impact sets?*

From our answer to the first research question (RQ 1) we realize that we can significantly improve the impact set detection accuracy of Tarmaq through analyzing regular and micro-clones. However, we still do not know whether regular code clones or micro clones have more contributions towards improved detection of impact sets. We investigate this matter in this section and answer the second research question (RQ 2) by analyzing our experimental results.

Investigation procedure. For answering RQ 2, we define and compare the impact set detection accuracies of the following two variants of Tarmaq-NiCad.

(1) Tarmaq-NiCad-Regular (TNR): This variant only considers the regular code clones detected by NiCad for detecting the impact sets. The minimum size of a regular clone fragment is 5LOC as was suggested by Wang et al. [46].

(2) Tarmaq-NiCad-Micro (TNM): This variant only considers the micro-clones detected by the NiCad clone detector for retrieving the impact sets. Micro-clones can be of at most 4LOC as was considered in the literature [33].

We determine the precisions and recalls of these two techniques by following the procedure described in Section VI-A. In the following paragraphs, we make a comparison between these two techniques.

Comparing the variants. Fig. 5 and Fig. 6 compare the precisions and recalls of the two techniques: Tarmaq-NiCad-Regular (TNR), and Tarmaq-NiCad-Micro (TNM). From Fig. 5 we see that the precision regarding TNR is always higher than that of TNM. However, Fig. 6 shows that the recall of TNM is mostly higher (for 6 subject systems out of 8) than the recall of TNR except for Camellia and DNSJava. For each of these two subject systems, the recall of TNR is slightly higher than that of TNM.

The reason behind higher precision of TNR and mostly higher recall of TNM is that micro-clones are generally bigger in number than regular code clones. When we consider micro-clones, many false positives get introduced and the precision of TNM drops below the precision of TNR. However, micro-clones increase the number of true positives as well, and it makes the recall of TNM to be mostly higher than TNR.

Findings from our analysis. According to our investigation and analysis, regular clones appear to be more beneficial than micro-clones when we consider improvement in precision. However, micro-clones often appear to be more beneficial than regular clones when we consider improvement in recall in detecting impact sets.

C. Answering the Third Research Question (RQ 3)

RQ 3: *Which sizes of micro-clones are more beneficial towards improving the detection accuracy of impact sets?*

TABLE IV
NUMBER OF TRUE POSITIVES OBTAINED BY ANALYZING DIFFERENT SIZES OF MICRO-CLONES

Systems	Size = 1LOC	Size = 2LOC	Size = 3LOC	Size = 4LOC
Ctags	6	292	318	74
Camellia	2	0	138	108
Brlcad	4	2	138	966
Carol	0	30	1014	454
Freecol	94	4	2328	558
DNSJava	38	100	1338	1610
Greenshot	12	94	936	248
MonoOSC	0	8	224	272

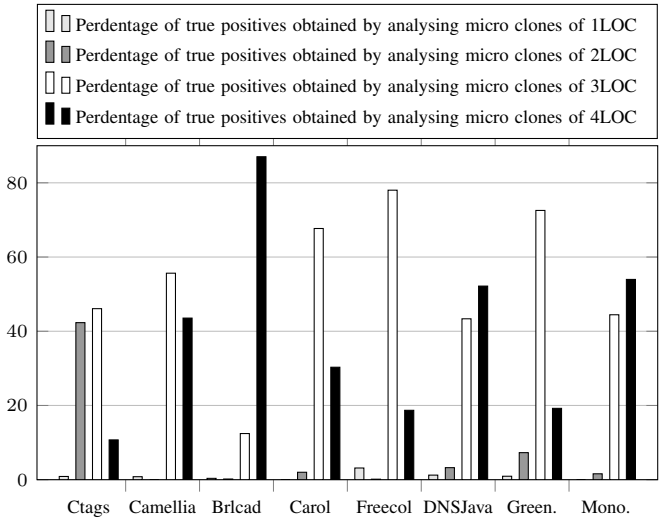

Fig. 7. Percentage of true positives obtained from different sizes of micro-clones

From our answer to RQ 2 we realize that micro-clones are generally more beneficial than regular clones towards improving the recall in detecting impact sets. In this subsection, we make a comparison among different sizes of micro-clones on the basis of their contributions towards improving the recall. If the contribution from the micro-clones of a particular size appears to be very low, we can discard those micro-clones from our consideration. We answer RQ 3 in this section through our investigation and analysis.

Investigation procedure. For answering RQ 3, we measure the contributions of different sizes of micro-clones towards improving the detection accuracy of impact sets. Micro-clones are of four different sizes: 1, 2, 3, or 4LOC. Considering the micro-clones of each size, we determine the number of true positives co-change candidates that we can retrieve by analyzing those micro-clones only. The number of true positive co-change candidates from each size of micro-clones is shown in Table IV. We see that the number of true positives obtained from micro-clones of size 3LOC and 4LOC are mostly higher than the number of true positives retrieved from micro-clones of 1LOC and 2LOC.

Considering the micro-clones of a particular size, we also determine the percentage of true positives that we can obtain from those micro-clones with respect to all true positives obtained from all four sizes of micro-clones. These percentages are shown in Fig. 7. The figure again shows that the percentages regarding 3LOC and 4LOC micro-clones are mostly higher than the percentages regarding 1LOC and 2LOC micro-clones except for Ctags. In the case of Ctags, the percentage of true positives obtained from 2LOC micro-clones is higher than the corresponding percentage regarding 4LOC micro-clones. Finally, it seems that micro clones of 3LOC and 4LOC are more beneficial towards improving the detection accuracy of impact sets compared to the micro-clones of other sizes.

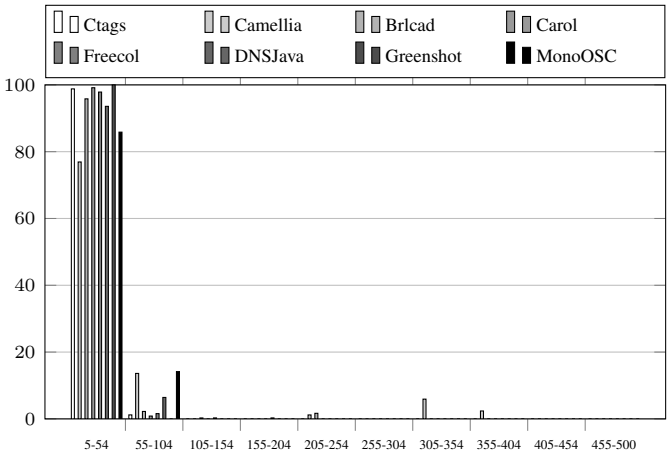

Fig. 8. Possibilities of getting true positive clone fragments at different ranges of clone sizes

Findings from our analysis: According to our experimental results, when detecting impact sets, micro-clones of 3LOC and 4LOC should be prioritized more compared to the micro-clones of less than 3LOC.

D. Answering the Fourth Research Question (RQ 4)

RQ 4: *Which sizes of regular clones are more beneficial for improving the detection accuracy of impact sets?*

The minimum and maximum size thresholds of regular clones are respectively 5LOC and 500LOC in our research. We wanted to determine if the regular code clones of a particular size range mostly provide the true positive co-change candidates during impact analysis. If such a size range really exists, we can ignore the regular code clones beyond that size range. The clone detector can be set to detect regular clones of that particular size range excluding the clones beyond the range. Such an exclusion can also make the clone detection process faster for the purpose of change impact analysis. We perform our investigation in the following way.

Investigation procedure. Let us assume that we are now determining the impact set for a particular target method m through clone analysis. We detect code clones from the entire code-base using the NiCad clone detector. Let us again assume that the target method m intersects a particular clone fragment in a clone class. We consider the other clone fragments in the

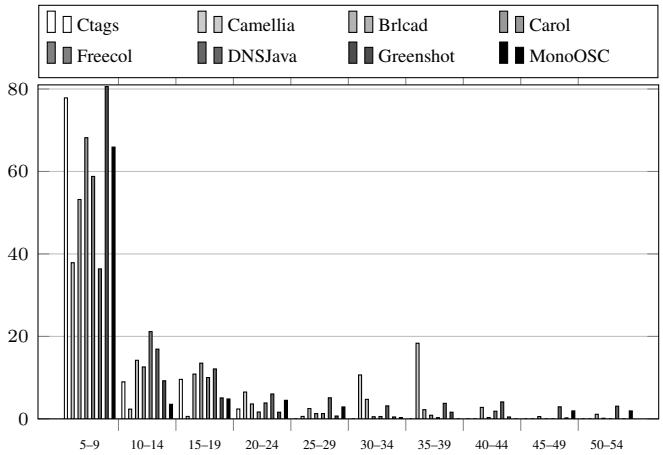

Fig. 9. Possibilities of getting true positive clone fragments at different intervals of clone sizes in the range 5 to 55

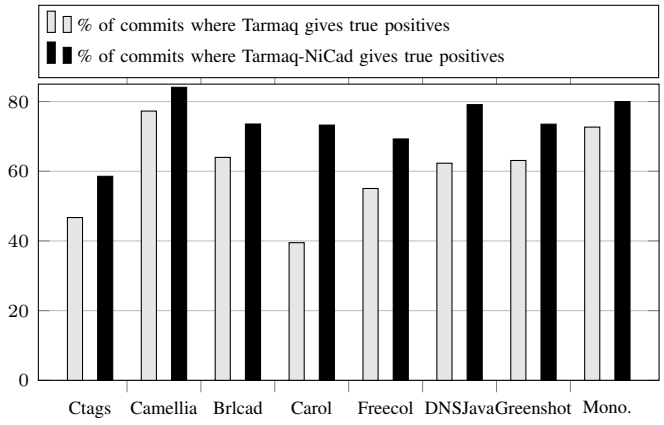

Fig. 10. Comparing the percentages of commits where Tarmaq and Tarmaq-NiCad can provide impact sets with true positives

class as a set, OCF (other clone fragments). We determine the methods that are intersected by the members in OCF. Let us consider that some of these intersected methods are true positives (i.e., actually co-changed with the target method m). We identify which members in OCF intersected the true positive methods. These identified members (i.e., identified clone fragments) in OCF are the true positive members, that is, **true positive clone fragments**. Thus, a **true positive clone fragment** is a clone fragment which is detected by NiCad and which intersects with a true positive co-change candidate (i.e., a true positive method).

For the entire period of evolution of a subject system, we determine all the true positive clone fragments for all the target methods. For each such clone fragment, we determine the size of it in LOC. Finally, considering all the true positive clone fragments from all the target methods from the whole period evolution of each of our subject systems, we determine the possibilities of getting true positive clone fragments at different size intervals. Let us explain this using an example.

Let us assume that there are five true positive clone fragments during the entire period of evolution of a software system. The sizes of these true positive fragments are: 5, 5, 10, 8, and 7. Let us now determine the possibilities of getting true positives in two intervals: 5 to 7 and 8 to 10. This is easy to determine that the possibility (in percentage) for the first interval (5 to 7) is 60% (=3×100/5). We see that three true positive clone fragments out of five have sizes within this interval. In the same, way the possibility regarding the third interval (8 to 10) is 40% (=2×100/5).

As we have done in the above example, we determine the possibilities of getting true positive clone fragments in different size intervals as is depicted in Fig. 8. We see that most of the intervals are of 50LOC except the last interval which is of 45LOC. As we set the maximum size threshold of the clone detector to be 500, we fix the last interval to be in the range 455 to 500. However, we see that the possibility regarding this interval is 0% for all the subject systems.

According to Fig. 8, the possibilities regarding the first interval is very high (around 100%) for each of the subject systems. The possibilities for the other intervals are mostly zero except for the second and seventh intervals of Camellia and the second interval of DNSJava and MonoOSC. For the second intervals of Camellia, DNSJava and MonoOSC, the possibilities are respectively 13.61%, 6.42%, and 14.15%. For the seventh interval ([305 - 354]) of Camellia, the possibility is around 5.92%. Such a scenario implies that for the purpose of detecting co-change candidates, regular code clones within the size range of 5LOC to 55LOC are enough. Regular clones of all other ranges have a negligible possibility of appearing as true positive clone fragments.

As the possibility within the first interval [5 - 55] is very high for each of our subject systems, we wanted to further investigate how this possibility is distributed over this interval. Fig. 9 shows this distribution. In this figure, we can see the probabilities of getting true positives in different ranges of clone size. Each range is of 5LOC. We see that the possibility regarding the first range ([5 - 9]) is the highest for each of the subject systems. The possibilities after the 3rd range ([15 - 19]) are very low (mostly less than 10). Such a scenario of probability distribution implies that we primarily need to focus on detecting regular clones of 5LOC to 20LOC for the purpose of change impact analysis.

Findings from our analysis. According to our experimental results and analysis, regular code clones within the range of 5LOC to 20LOC are the most beneficial ones for change impact analysis.

E. Answering the Fifth Research Question (RQ 5)

RQ 5: *Does a combination of clone analysis and evolutionary coupling detect impact sets more often compared to evolutionary coupling alone?*

We finally investigate whether consideration of code clones can often help us improve the impact sets predicted by evolutionary coupling based techniques. Our investigation in

TABLE V
STATISTICS REGARDING THE NUMBER OF COMMITS WHERE TARMAQ OR TARMAQ-NICAD PROVIDED IMPACT SETS WITH TRUE POSITIVES

Measure	Ctags	Camellia	Brlcad	Carol	Freecol	DNSJava	Greenshot	Mono.
No. of commits where more than one method got changed	152	88	261	243	485	613	336	150
No. of commits where Tarmaq provided true positives	71	68	167	96	267	382	212	109
No. of commits where Tarmaq-NiCad provided true positives	89	74	192	178	336	485	247	120

this section answers the fifth research question (RQ 5). We investigate in the following way.

Investigation procedure. For answering RQ 5, we apply each of the two impact set detection techniques, Tarmaq and Tarmaq-NiCad, on each of the revisions of our subject systems as we did for answering RQ 1, and determine three measures: (i) the total number of commits where more than one method got changed, (ii) the number of commits where Tarmaq provided impact sets with true positives, and (iii) the number of commits where Tarmaq-NiCad provided impact sets with true positives. Table V shows these measures for each of our subject systems. Fig. 10 shows the percentages of commits where Tarmaq and Tarmaq-NiCad provided true positives with respect to all commits that affected two or more methods. We see that the percentage regarding Tarmaq-NiCad is always higher than that of Tarmaq. According to Wilcoxon Signed Rank test [48], [49] results, the percentages regarding Tarmaq-NiCad are significantly higher than those of Tarmaq with a p-value of 0.01 for two-tailed test case.

Findings from our analysis. According to our investigation, combination of clone analysis with Tarmaq can help it provide impact sets with true positives for a significantly higher number of commit operations during the whole period of system evolution.

Our finding from RQ 5 again establishes the importance of considering code clones for change impact analysis.

VII. THREATS TO VALIDITY

We have used the clone detection tool called NiCad [23] for our investigation. For different settings of NiCad, the clone detection results can be different. Wang et al. [46] mention it as a *confounding configuration choice problem*. However, the settings that we have used for detecting regular clones are considered standard. Recently Svajlenko and Roy [24] used these settings for comparing NiCad with other modern clone detectors and found NiCad to be a promising choice in terms of precision and recall for detecting clones. Moreover, the settings that we have used for detecting micro-clones were previously used by Mondal et al. [33]. We finally believe that the experiment results and findings that we have got from our research are of significant importance.

We did not investigate enough subject systems in our experiment. Thus, our findings might not be generalized. However, the systems that we have analyzed are of diverse variety in terms of their application domains, implementation languages, sizes, and revision history lengths. Thus, our findings cannot be attributed to a chance. We believe that these findings can be important for an improved detection of impact sets considering method granularity during change impact analysis.

The number of revisions that we have investigated for different subject systems might not be enough to establish our findings. However, from Table II we see that different subject systems have different lengths of revision history and these lengths span from small to large. We thus believe that our findings are not biased by the revision history lengths of the subject systems.

VIII. CONCLUSION

In this paper we investigate whether consideration of code clones with evolutionary coupling can help us in better change impact analysis compared to evolutionary coupling alone. For the purpose of our investigation, we select the state-of-the-art technique called Tarmaq that can find impact sets for a target program entity by analyzing evolutionary coupling. We combine clone analysis with Tarmaq and investigate whether the combined technique can provide us impact sets with better accuracy (better precision and recall) than the original technique. According to our investigation on thousands of commits of eight diverse subject systems, clone analysis can significantly improve the precision and recall of Tarmaq in predicting impact sets considering method level granularity. According to our analysis regarding clone size, micro-clones of 3LOC and 4LOC and regular clones within the range of 5LOC to 20LOC contribute the most towards improving the detection accuracy of impact sets. We also observe that consideration of code clones with evolutionary coupling can help us predict impact sets with true positives for a significantly higher percentage of commit operations compared to evolutionary coupling alone. We expect that clone analysis can also be used with other impact analysis techniques and measures for a better analysis of software change impacts. We will investigate this in future.

ACKNOWLEDGEMENT

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by two Canada First Research Excellence Fund (CFREF) grants coordinated by the Global Institute for Food Security (GIFS) and the Global Institute for Water Security (GIWS).

REFERENCES

[1] A. Alali, B. Bartman, C. D. Newman, J. I. Maletic, "A Preliminary Investigation of Using Age and Distance Measures in the Detection of Evolutionary Couplings", in Proc. *MSR*, 2013, pp. 169 – 172.

[2] A. Lozano, M. Wermelinger, "Assessing the effect of clones on changeability", Proc. *ICSM*, 2008, pp. 227 – 236.

[3] A. Lozano, M. Wermelinger, "Tracking clones' imprint", Proc. *IWSC*, 2010, pp. 65 – 72.

[4] C. K. Roy, "Detection and analysis of near-miss software clones", Proc. *ICSM*, 2009, pp. 447 – 450.

[5] C. K. Roy, J. R. Cordy, "A Mutation / Injection-based Automatic Framework for Evaluating Code Clone Detection Tools", Proc. *Mutation*, 2009, pp. 157 – 166.

[6] C. K. Roy, J. R. Cordy, "A Survey on Software Clone Detection Research", *Technical Report No. 2007-541*, 2007, School of Computing Queen's University, pp. 1 - 115.

[7] C. K. Roy, M. F. Zibran, R. Koschke, "The Vision of Software Clone Management: Past, Present, and Future (Keynote paper)", Proc. *CSMR-WCRE*, 2014, pp. 18 – 33.

[8] C. Kapser, M. W. Godfrey, ""Cloning considered harmful" considered harmful: patterns of cloning in software", *Empirical Software Engineering*, 2008, 13(6): 645 – 692.

[9] CTAGS: http://ctags.sourceforge.net/.

[10] D. Poshyvanyk, A. Marcus, R. Ferenc, T. Gyimóthy, "Using Information Retrieval based Coupling Measures for Impact Analysis", *ESE*, 2009, vol. 14, no. 1, pp. 5 – 32.

[11] D. Poshyvanyk, A. Marcus, "The Conceptual Coupling Metrics for Object-Oriented Systems", Proc. *ICSM*, 2006, pp. 469 – 4.78.

[12] E. Juergens, F. Deissenboeck, B. Hummel, S. Wagner, "Do Code Clones Matter?", Proc. *ICSE*, 2009, pp. 485 – 495.

[13] F. Bantelay, M. B. Zanjani, H. Kagdi, "Comparing and Combining Evolutionary Couplings from Interactions and Commits", in Proc. *WCRE*, 2013, pp. 311 – 320.

[14] F. Jaafar, Y. Gueheneuc, S. Hamel, G. Antoniol,"An Exploratory Study of Macro Co-changes", Proc. *WCRE*, 2011, pp. 325 – 334. .

[15] G. Antoniol, G. Canfora, G. Casazza, and A. Lucia, "Identifying the Starting Impact Set of a Maintenance Request: A Case Study", Proc. *CSMR*, 2000, pp. 227 – 231.

[16] G. Bavota, B. Dit, R. Oliveto, M. D. Penta, D. Poshyvanyk, A. D. Lucia, "An Empirical Study on the Developers' Perception of Software Coupling", Proc. *ICSE*, 2013, pp. 692–701.

[17] G. Canfora, L. Cerulo, "Impact Analysis by Mining Software and Change Request Repositories", Proc. *METRICS*, 2005, pp. 20 – 29.

[18] G. Canfora, M. Ceccarelli, L. Cerulo, M. D. Penta,"Using Multivariate Time Series and Association Rules to Detect Logical Change Coupling: an Empirical Study", Proc. *ICSM*, 2010, pp. 1 – 10.

[19] H. Gall, M. Jazayeri, J. Krajewski,"CVS Release History Data for Detecting Logical Couplings", Proc. *IWPSE*, 2003, pp. 13 – 23.

[20] H. Kagdi, M. Gethers, D. Poshyvanyk, M. L. Collard,"Blending Conceptual and Evolutionary Couplings to Support Change Impact Analysis in Source Code", Proc. *WCRE*, 2010, pp. 119 – 128.

[21] J. Krinke, "A study of consistent and inconsistent changes to code clones", Proc. *WCRE*, 2007, pp. 170 – 178.

[22] J. Law, G. Rothermel, "Whole Program Path-Based Dynamic Impact Analysis", Proc. *ICSE*, 2003, pp. 308 – 318.

[23] J. R. Cordy, C. K. Roy, "The NiCad Clone Detector", Proc. *ICPC Tool Demo*, 2011, pp. 219 – 220.

[24] J. Svajlenko, C. K. Roy, "Evaluating Modern Clone Detection Tools", Proc. *ICSME*, 2014, pp. 321 – 330.

[25] K. Hotta, Y. Sano, Y. Higo, S. Kusumoto, "Is Duplicate Code More Frequently Modified than Non-duplicate Code in Software Evolution?:

[28] L. Briand, J. Wust, and H. Louinis, "Using Coupling Measurement for Impact Analysis in Object-Oriented Systems", Proc. *ICSM*, 1999, pp. 475 – 482.

An Empirical Study on Open Source Software", Proc. *EVOL/IWPSE*, 2010, pp. 73 – 82.

[26] L. Aversano, L. Cerulo, and M. D. Penta, "How clones are maintained: An empirical study", Proc. *CSMR*, 2007, pp. 81 – 90.

[27] L. Barbour, F. Khomh, Y. Zou, "An empirical study of faults in late propagation clone genealogies", *Journal of Software: Evolution and Process*, 2013, 25(11):1139 – 1165.

[29] M. A. Islam, M. M. Islam, M. Mondal, B. Roy, C. K. Roy, K. A. Schneider, "Detecting Evolutionary Coupling Using Transitive Association Rules", Proc. *SCAM*, 2018, pp. 113 – 122.

[30] M. Beller, A. Zaidman, A. Karpov, "The Last Line Effect", Proc. *ICPC*, 2015, pp. 240 – 243. .

[31] M. D'Ambros, M. Lanza,"Reverse Engineering with Logical Coupling", Proc. *WCRE*, 2006, pp. 189 –198.

[32] M. Kim, V. Sazawal, D. Notkin, G. C. Murphy, "An empirical study of code clone genealogies", Proc. *ESEC-FSE*, 2005, pp. 187 – 196.

[33] M. Mondal, C. K. Roy and K. A. Schneider, "Micro-clones in evolving software," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, 2018, pp. 50 – 60.

[34] M. Mondal, C. K. Roy, K. A. Schneider, "An Empirical Study on Clone Stability", *ACM SIGAPP Applied Computing Review*, 2012, 12(3): 20 – 36.

[35] M. Mondal, C. K. Roy, K. A. Schneider, "Automatic Ranking of Clones for Refactoring through Mining Association Rules", Proc. *CSMR-WCRE*, 2014, pp. 114 – 123.

[36] M. Mondal, C. K. Roy, K. A. Schneider, "Prediction and Ranking of Co-change Candidates for Clones", Proc. *MSR*, 2014, pp. 32 – 41.

[37] M. Mondal, C. K. Roy, K. A. Schneider,"Improving the detection accuracy of evolutionary coupling by measuring change correspondence", Proc. *WCRE-CSMR*, 2014, pp. 358 –362. .

[38] M. Mondal, C. K. Roy, K. A. Schneider,"Improving the Detection Accuracy of Evolutionary Coupling", Proc. *ICPC*, 2013, pp. 223 – 226.

[39] M. Mondal, C. K. Roy, M. S. Rahman, R. K. Saha, J. Krinke, K. A. Schneider, "Comparative Stability of Cloned and Non-cloned Code: An Empirical Study", Proc. *SAC*, 2012, pp. 1227 – 1234.

[40] N. Göde, J. Harder, "Clone Stability", Proc. *CSMR*, 2011, pp. 65 – 74.

[41] Online SVN repository: http://sourceforge.net/.

[42] R. Agrawal, T. Imielinski, A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *ACM SIGMOD*, 1993, 22(2):207 – 216.

[43] R. Robbes, D. Pollet, M. Lanza,"Logical Coupling Based on Fine-Grained Change Information", Proc. *WCRE*, 2008, pp. 42 – 46.

[44] R. van Tonder, C. Le Goues, "Defending against the attack of the micro-clones", Proc. *ICPC*, 2016, pp. 1 – 4. .

[45] T. Rolfsnes, S. D. Alesio, R. Behjati, L. Moonen and D. W. Binkley, "Generalizing the Analysis of Evolutionary Coupling for Software Change Impact Analysis", Proc. *SANER*, 2016, pp. 201 – 212.

[46] T. Wang, M. Harman, Y. Jia, J. Krinke, "Searching for Better Configurations: A Rigorous Approach to Clone Evaluation", Proc. *ESEC/SIGSOFT FSE*, 2013, pp. 455 – 465.

[47] T. Zimmermann, P. Weisgerber, S. Diehl, A. Zeller, "Mining version histories to guide software changes", Proc. *ICSE*, 2004, pp. 563–572.

[48] Wilcoxon Signed Rank Test. https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test.

[49] Wilcoxon Signed Rank Test. http://www.socscistatistics.com/tests/signedranks/Default2.aspx.

[50] Effect Size and Interpretation: https://en.wikipedia.org/wiki/Effect_size.

[51] Wilcoxon Signed Rank Test and Cohen's d effect size calculator: https://www.ai-therapy.com/psychology-statistics/effect-size-calculator.

[52] S. Pugh, D. Binkley, and L. Moonen. The case for adaptive change recommendation. In *SCAM*, pages 129 – 138, 2018.