International Journal of Computational Intelligence and Informatics, Vol. 5: No. 4, March 2016

A Review on Code Smell Techniques Using Nesting Structure Tree

Chaitanya Kulkarni
Computer Department
JJT,University
Jhunjhunu,India
Ckulkarni47@gmail.com

S. D.Joshi
Computer Department
Bharati Vidyapeeth
Pune,India
sdj@live.in

***Abstract*-Duplicated code is harmful for maintaining and evolving the source code in software systems. The refactorability supports to remove the software clones and checking the unification while merging the duplicated code. An approach for automatically pair the clones that can be refactored without changing the behavior of program is needed. There is a problem by using the support of removing the clones from the source code through refactoring because they are having more modification in the code that may not support. The proposed approach will examine differences between clones that can be safely parameterized without causing any side-effects. The computational cost of the proposed approach is negligible (less than a second) in the huge majority of the examined outcomes. The clones will be detected by tools which are discussed by clone types. Input to detect the clones is source code which consists of set of statements and preconditions. By using Program Dependence Graph (PDG) mapping and Abstract Syntax Tree (AST) matching techniques can detect clones from source code with the help of refactoring technique. Using variety of clone detection tools on open source project the present work's performance will be studied and also assessment of refactorable clones in production code is more than the clones in test code there will be done.**

Keywords- Software Clone, Code Clone Management, Refactoring of code clone.

I. Introduction

Refactorability is the process of changing the external behavior of the source code without altering the internal system of the software. The main role of the refactoring is to identify the duplicated code in a program. The clone is the similar item of the code which is used more than the same statements, preconditions and same methods. The refactorability can refactor the clones in a source code but the clones are not refactored completely by using the refactoring technique. The code replication is a potentially serious problem having an unnecessary impact on the software maintainability, comprehensibility and progress of the software systems [1]. The tools of the clone detection tools are refactored automatically to remove the software clones and the refactorability analysis is important to find the duplicated code in the source code and the maintainace of the source code is important in clone management which is to filter the clones and refactored directly [2]. The developer and maintainer will refactor and maintain the refactored code throughout the software development. The approaches which overcome the refactoring through clones first is to find the code fragments with the nesting structures and later combine the two code fragments which is used to refactor the duplicated code from the present source code. Next is to find the statements which are same in the two code fragment and then the statements are mapped if they are same if not they are unmapped statements. From these mapped and unmapped statements the differences are displayed and they are highlighted by yellow color. The refactoring will support the mapping results with the minimum number of differences between mapped statements and the compares the refactorability which is the alternative of the mapped results in a maximum number of differences. There are some differences which cannot refactor safely and they lead to the differences which may increase the side effects of the parameterization. The difference between the mapped and unmapped statements is examined against the set of preconditions which are determined whether they are parameterized without changing the behavior of the source code. Previously the control dependence tree is used and that is replaced with program dependence tree which is the (SESE) single entry and single exit of the every node which is used to build the source code [3]. The program dependence graph is used the directed graph with multiple edges which represent the control and flow dependencies between the control predicate statements and non-predicate statements The conditions is divide and conquer algorithm is used to get the results when two source code fragment is combined and they are safely refactored. To the statement mapping problem the statements are combined and check whether they matched are not [1]. Later the matched statements are listed as a preconditions and they detect the precondition violations after extracting the methods from the source code and the extracted code is checking the statements and preconditions and they are extracted in one separate method which specified by the user. Or crosswise over various projects possessed or kept up by some amount. Copy code is viewed as undesirable, for various reasons.

II. Proposed Approach

An approach of these is the process of single as well as double forms. There are many code fragments within the same body as different method is same methods that are reported as clones. The extracted two code fragments are having the same methods and same statements which contain duplication. By using the nesting structure tree matching, statement mapping, preconditions to refactorable the clones and every code fragment having the number of statements and preconditions which contain duplicated code.

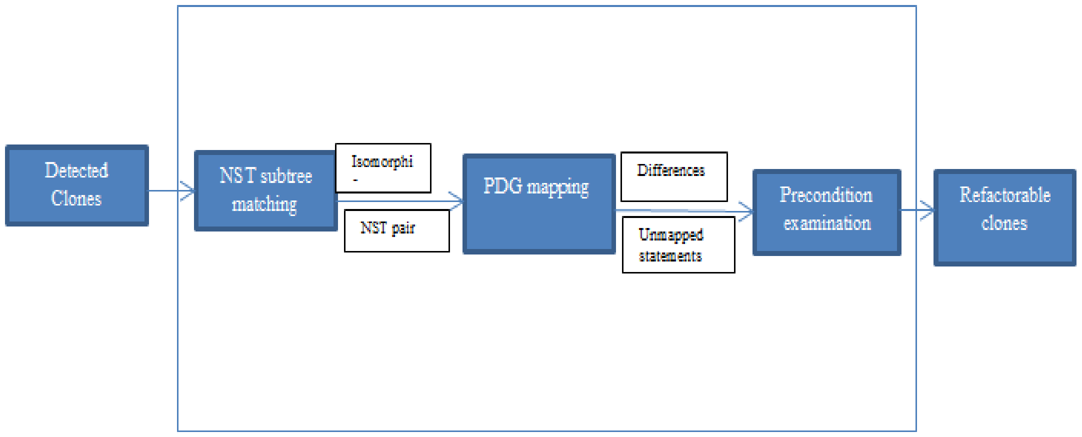

Figure 1. Proposed system architecture

Detected clones are two code fragments which is used to detect the clones from that two code fragments and they are parsing to the nesting structure tree matching to detect the clone pairs and match the exactly paired match and the paired match is parsing to the PDG mapping in these the statements are mapped each other's and shows the differences of every statement and the unmapped statements are parsing to precondition examination the preconditions are based on the differences between mapped statements and as well as unmapped statements regarding the behavior of source code is examined. The abstract syntax tree compatibility is the control predicate nodes which are used to match the AST compatibility structure of their conditional expressions. The two statements are compatible and they are corresponding to the same AST compatible statements having the isomorphic AST structure [4]. Nesting structure tree is the proposed technique to find the maximum isomorphic subtree within the NST matching. Code fragments having the identical nesting tree and initially the two fragment are collected two NST's of all leaf nodes. The sibling nodes having the clone pairs which is given as input to the matching tree. Statement mapping is an algorithm which extracts the subtrees of the NST's to give the input of the code fragment. By using the statement mapping the divide and conquer method is used to refactor the clones and by using these statements are divided into two code fragments and they are matched the pair using NST's and pairs the node. These extracted code may highlighted by using white color. The rename of the method is used and we can rename the internal structure of the code. Precondition is the used to know whether the code fragments can be refactored safely and they are integrated into one common method. The preconditions are handling the difference between mapped and unmapped statement the unmapped statements are directly refactored without any error and they are handled easily. The existing approach is Clone ranking techniques for the purpose of refactoring uses the method of combining clone metrics to extract code clones. It analyzes the differences between the clones and displays the reason for clones that are not refactorable [5]. In some other cases it helps the developers to make the clones refactorable. These techniques automate the application of refactoring, when there are no precondition violations. The drawback of these increases the maintainace cost and introduce the new bug is propagating to the bugs which overcome the safely refactored source code.

III. Literature Survey

According to the C.K.Roy the refactorability of software clones having the several clone detection tools and techniques in which they are identified by the clones in a code which are influenced and forced by the developer and maintenance engineering [6] The software system itself do not occur the clones by themselves but they are introduced by an accident. The literature may contain many approaches which are discussed and the duplication of code in software is refactored safely. There is a drawback in the code clones is that clones in a code will change the code fragment in a same code to overcome the automatic refactoring is developed. Therefore the refactoring doesn’t improve the software always with a code clones and software clone detection and refactoring both will overcome the problem to detect the clones from source code because in a software program there are different kinds of clones which occurs then the refactoring may applied to that source code to eliminate the clones from a program [7]. The duplicated code is seems to be reasonable or beneficial design where it is correct and useful detection of clones in a code and then refactoring will support always and they also face some problems to remove the clones from program. According to Elmar Juergens [8] the code clones are essential to remove from the program which are affected by the software that may uses the detection of inconsistent clones to identify the clones and remove the clones by tools which are provide. According to Angela Lozano [6] the effects of clones are changing due to the similar code fragments in the source code and they are assessed by using the clone detection tools without changing the behavior of program. The clones are harmful and complex to understand and they lead to the incomplete updates to generate the bugs. The post and pre conditions are the refactoring techniques which will group the clone and the remove the invariants from them [9].

IV. CONCLUSION

Refactorability is the technique which is used to maintain the cost and also we can reuse the code anywhere without any changes and the refactorability is one of the easy technique which can be safely refactor and the comparison of the two code fragments is used to develop the best refactorable code in the form of divide and conquer fashion and the techniques are developed using the nesting structure, statement mapping and preconditions these will be performed by using the code fragments and it maintains the cost and complex also and the future work of the is used to refactor more than two code fragments and checks the unification of those code.

V. FUTURE SCOPE

Nesting structure tree mechanism will helps to assess refactored of code clones fragments. It will also support for refactoring of clones using tree mechanism. It will help us to refactoring of code to find maximal level of depth solution.

VI. ACKNOWLEDGEMENT

The author would like to thank the faculty of the JJT, University computer dept. And also I would like to thank you guide Dr.S.D.Joshi professor Bharati Vidyapeeth, Pune, to give novel idea in code clone management system which is useful commercially in software engineering.

REFERENCES

[1] Nikolaos Tsantalis, Davood Mazinanian, and Giri Panamoottil Krishnan, “Assessing the Refactorability of software Clones,” in Proc. IEEE Transactions on software engineering, vol.41, no.11, November 2015.

[2] H. A. Nguyen, T. T. Nguyen, N. H. Pham, J. Al-Kofahi, and T. N. Nguyen, “Clone management for evolving oftware,” IEEE Trans. Softw. Eng., vol. 38, no. 5, pp. 1008–1026, Sep.-Oct. 2012.

[3] R. Johnson, D. Pearson, and K. Pingali, “The program structure tree: Computing control regions in linear time,” in Proc. ACM SIGPLAN Conf. Programm. Lang. Des. Implementation, pp. 171–185, 1994.

[4] P. Hell and J. Nesetril, Graphs and Homomorphism’s (series Oxford Lecture Series in Mathematics and Its Applications). London, U.K.: Oxford Univ. Press, 2004.

[5] Roy C.K. and Cordy J.R., “A Survey on Software Clone Detection Research”,Queen’s School of Computing, Technical Report No.2007-541, vol.115, September 2007.

[6] C. K. Roy, J. R. Cordy, and R. Koschke, “Comparison and evaluation of code clone detection techniques and tools: A qualitative approach,” Sci. Comput Programm., vol. 74, no. 7, pp. 470–495, 2009.

[7] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, E. Merlo, Comparison and evaluation of clone detection tools, Transactions on Software Engineering 33 (9), 577_591, 2007.

[8] Arcelli Fontana, Francesca, et al. "Software clone detection and refactoring.*"ISRNSoftware Engineering* 2013.

[9] C. Roy, M. Zibran, and R. Koschke, “The vision of software clone management (Past, present, and future (keynote paper),” in Proc. IEEE Conf. Softw.Maintenance, Reeng. Reverse Eng., Softw. Evol.Week, pp. 18–33, 2014.

[10] G. P. Krishnan and N. Tsantalis, “Unification and refactoring of clones,” in Proc. IEEE Conf. Softw. Maintenance, Reeng. Reverse Eng. Softw. Evol. Week, pp. 104–113, 2014.