IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-6, Special Issue-3, April 2018 E-ISSN: 2347-2693

A Survey on Software Code Clone Detection to Improve the Maintenance Effort and Maintenance Cost of the Software

V. Guna[1*], M. Sunil Kumar[2]

[1*]Department of Computer Science and Engineering, Sree Vidyanikethan Engineering, Tirupati, India
[2] Department of Computer Science and Engineering, Sree Vidyanikethan Engineering, Tirupati, India

[*]*Corresponding Author: sunilmalchi@gmail.com*

Available online at: www.ijcseonline.org

Abstract— During the development of the software the developers have a chance to copy the code continuously. Due to copying of the code there is a chance of having the identical or more similar code fragments in the software and it is called as software clones or code clones. These clones can be detected from the existing code that is in c, c++, java etc programming languages. By the Argo UML tool to the existing code to generate the class diagrams by using reverse engineering process. In software development process, coping of existing code fragment and pasting them with or without modification is a frequent process. Code clone means copy of an original form or duplicate. Software clone detection is important to reduce the software maintenance cost and to recognize the software system in a better way. There are many software code clone detection techniques such as text- based, token-based, Abstract Syntax tree based etc. and they are used to spot and finding the existence of clones in software system. Mainly detection of clones is on the type-1, type-2 and type-3 clones. These clones can be detected by using several novel algorithms are ARIMA, Back propagation, Multi objective genetic algorithm, support vector machines and also with several hybrid techniques with respect to recall and precision.

Keywords— Code Clones, Software maintenance, Type-1, Type-II and Type-III clones, Recall and Precision

I. INTRODUCTION

Definition 1: Code Fragment. A code fragment (CF) is any sequence of code lines (with or without comments). It is a sequence of statements.

Definition 2: Software Clone. A code fragment is a clone of another code if they are similar

In general, clones are set of identical segments of code in a software system, which has a bad impact in the system. In software development approach, duplicating previous code segments in different programs with or without modification is frequent process and that duplicated code is extremely difficult to maintain. The imitation in code is known as software clone and the phenomenon is known as software cloning. Code Cloning considered as a bad smell in software industry and has a bad impact on software quality, software maintenance and also increases maintenance cost. Roy and Cordy mentioned software clone as software reuse. Although, it is a fast and instant method of software reuse yet, it is a harmful design procedure.

Code reuse is a standard practice in modern software development. The downside of code reuse via replication and copy-paste programming is that it leads to code bloat, increasing the technical depth of software products and making maintenance costly and time consuming

During the software development process, the software engineer copy one piece of fragment and paste with or without alteration like renaming, addition or deletion etc to the fragment to reduce time and efforts. The process of coping and pasting of code fragment is known as code cloning and the copied pasted code is known as code clone. The reusing code is common practice in modern software developing, but it has some drawback It raises the maintenance cost while decreasing quality like changeability and updating of the software system. It also increase the chances of the bugs in the software system, because bug present in one fragment can be copied and pasted various time may increase the bugs in the software system

There are two types of similarities exist in the code clones one is syntactic similarities and other is semantic similarities. If the text of the clone code matches then it syntactic similarity and if the function or implementation of the code clone matches then it is semantic similarity. Code clone are of four types described below.

Type 1 – These are identical clones with syntactic similarities, in this type of clone only variation is allowed in whitespaces and comments.

Type 2 – These are renaming clones with syntactic similarities, in this type of clone only variation is allowed in literal, identifier, type, whitespaces and comments.

Type 3 – These are modified clones with syntactic similarities in this type of clone variation is allowed to rename literal, identifier and addition or deletion of statement to code.

Type 4 – These are semantic clone with semantic similarities, these clones are semantically same but syntactically different i.e. computation is same but implement by different syntactic variants.

Today, the software industry is getting more complex since the software systems are growing tremendously, so the software companies need a huge amount of the maintenance in terms of cost and efforts of existing software systems Software maintenance in software engineering is defined as the modification (corrective, adaptive, perfective, or preventative) of a software product after delivery to correct faults and improve the performance or other attributes various research studies have shown that maintenance of the software systems with code clones is more difficult than a non-cloned code system.

code clone is a code portion in source files that is identical or similar to another. Clones are introduced because of various reasons such as reusing code by ªcopy and paste,º mental macro (definitional computations frequently coded by a programmer in a regular style, such as payroll tax, queue insertion, data structure access, etc.), or intentionally repeating a code portion for performance enhancement

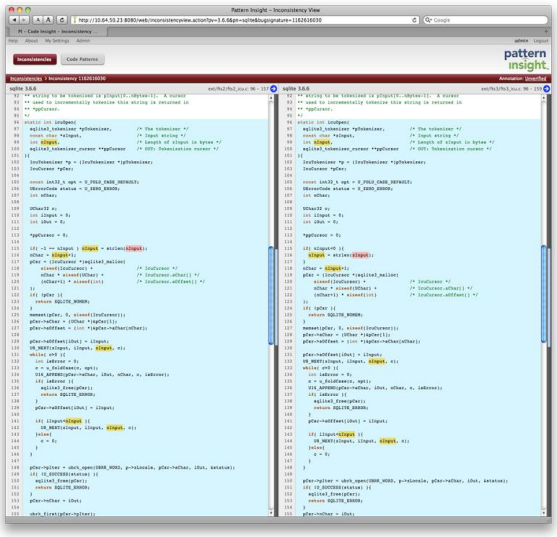

Figure 1: Example for software code clones

From the above figure the software clone clones are noticed that the marked part of the code is duplicated or repeated more than once in the code and that can behave in the same way or the different from each block of code.

II. LITERATURE SURVEY

In last decade many algorithm are proposed on software clone detection technique and every algorithm has its own advantage and disadvantage. This unit describes the summary and overview of recent research in the area of metric based software clone detection approach.

Y. Yuan et al. [1] proposed a count matrix based clone detection (CMCD) method, which is produced while counting the rate of frequencies of every variable in conditions specified by pre-determined counting condition. The projected technique is language-independent as it depends only on variable count. That is, if we have to count the rates of frequencies of variable in certain conditions with special standards, these standards are called as counting condition. Counting condition is used to select when the count should begin. The count matrix (CM) is a group of n count vectors (CV) and compares these Counting vectors with the help of Euclidean space. The variation between two vectors is calculated by the Euclidian Distance among them in the space, i.e.

$$D(V1,V2) = \left||V1 - V2|\right|2 = \sqrt{\sum_{i=}^{12} V2i)\text{^}2}$$

The CMCD perform well in extracting count-based information and it is language independent. It supports to detect clone in large programs (> 1M LoC) also it has a abilities to perform well in scenario-based evaluation.
Vidhya et al [1] proposed an emergent technique on java directories by using a metric based approach. The proposed system has been tested with two directories of JAVA files as input and the outcomes are produced based on the matching among files in directories. The percentage of the comparison is calculated by implementing the line by line comparison of the intermediate form of the files. This proposed technique merges both the textual based approach and metric based technique. Metric based approach is straight forward hence it is a light weight method. The textual based approach is the one which give high exactness. This proposed technique also helps to notice the directory level cloning that is not structurally correlated but functionally similar.

JAYADEEP PATI, BABLOO KUMAR, DEVESH MANJHI, AND K K SHUKLA [2] has proposed the different algorithms for clone detection during the software evolution. Most of the developers have the tendency to copy the modules in the programs completely or partially and modify them. It is about the evolution of clone components by using advanced time series analysis; clones

are extracted from the repository of the software by using abstract syntax tree approach. Then the analysis of evolution components is done it uses three models i.e. auto regressive integrated moving average, back propagation and multi objective genetic algorithm based on neural network, these techniques have been compared for the detection of the cloned components during the evolution of the software. The software evolution can be performed based on the large open source software application and Argo UML. This explains about the ability to predict the clones with the help of the software developers to reduce efforts during the software maintenance.

This involves in the identification of cloned components and also the prediction of the clone evolution in the open source software application. The identification of clones helpful in the field of bug prediction, it also helpful in reducing the corrective maintenance software clone evolution and prediction is helpful in perfective maintenance. MOGA-NN model is stated as the best model for predicting both types of clone number series. ARIMA model is for predicting the non-linear patterns in the data.

Stefan Bellon, Rainer Koschke [3] proposed the clone detection technique on text, lexical and syntactic information, software metrics and program dependency graphs. In this the detection of clones is based on two comparisons

1. Textual comparisons

2. Token comparisons
- It uses mapping algorithms

It uses Jens Krinke's tool it can able to analyze c systems only. All the other tools except this will handle both c and java. Krinke was not able to analyze the large programs in c, namely postgre sql, malthias rieger was not able to analyze postgre sql and the large java programs namely-J2sdk-1.4.0-javaX=swing. Token based technique and text based technique works damn similarly. The tools based on tokens and text have higher recall

There are several important points to note when looking at the results of the comparison:

1. The two token-based techniques and the text-based technique (Baker, Kamiya, and Rieger) behave astonishingly similarly. .

2. The tools based on tokens and text have higher recall.

3. Merlo's tool and Baxter's AST-based tool have higher precision.

4. The PDG-based tool (Krinke) does not perform too well (sensible only for type-3 clones). .

4. There is a large number of rejected candidates (between 24
Percent for Baxter and 77 percent for Krinke).

5. Many injected secret clones were missed (only between 24 percent and 46 percent of the injected secrets were found by the individual tools, ignoring Krinke who found only 4 percent because he analyzed only three of the eight programs).

The AST-based detection has a very high precision but currently has considerably higher costs in terms of execution time. The opposite is true for token-based techniques. If ideas from the token-based techniques could be made to work on ASTs, we would be able to find syntactic clones. with less effort. In fact, the Bauhaus project has developed a combined technique along these lines since then. The combined technique uses suffix-tree based recognition on serialized ASTs. The syntax-based technique could be improved if they took more advantage of their syntactic knowledge.

Elizabeth Burd, John Bailey [4] Proposed that the tools for detecting the code clones and also he shows the results of a process whereby the detection capacity of the 5 code replication detection tools. The aim is to remove some of the identical clones from the source code those tools are CCFinder, CloneDr, Covet, JPlag, Moss .

The result has no single and outright winner for clone detection for preventive maintenance. It can also identified the strengths and weakness in each tool that may ultimately lead to their improvement. Due to the plagiarism tools only considering across file duplication these are of less use than the duplicated clone detection tools and also it is possible to make more effective selection of clone identification tool.

Shruti Jadon [5] has stated that the clones can also increase the size of the program and creates the problem of redundancy. He is proposed to generate the feature sets after parsing the given C program for code fragments and then match their similarity on the basis of the feature sets the classification of algorithm is performed by using SVM as a machine learning tool. The output of this tool is similarity ratio within two C programs is related to each other and also the class in which they occur. Then by doing this it can increase its accuracy with the increase in number of instances. By having the code clones in the program increases the maintenance and also creates the problem of redundancy. The clone detection is in two stages in the first stage parser is used to generate the feature sets for this the

input is C file and the tool using is SVM. It is a machine learning tool to detect clones, this shows the accuracy increases with the increase in the number of instances. Then the final tool is used for the classification of input file as sorting and non-sorting class but this is limited only to the sorting class.

Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant Anna and Lorraine Bier [6] proposed the abstract syntax tree to the detection and removal of clones can decrease the software maintenance costs. It is used for detecting exact and near-miss clones over arbitrary program fragments in program source code by using abstract syntax trees. The tool using these techniques is applied to a C programs, thee tools produces macro bodies needed for clone removal, and macro innovations to replace the clones. This method determines the exact tree matches i.e a number of adjustments, commutative operands and nearly exact matches.

Chanchal K. Roy [7] proposed that clones are very harmful in the software maintenance and evolution. He develop Hybrid clone detection method and vagueness in clone detection by proposing a meta model of clone types then conducted a scenario based comparison and evolution of all currently available clone detection techniques and tools, in order to compare the available tools in a realistic setting, and also develop a mutation based framework that automatically and efficiently measures both recall and precision of the clone detection tool. Conducting a large scale empirical study of cloning in open source systems, and providing a scenario based comparison of the clone detection techniques and tools, to build a mutation based framework for automatically evaluating clone detection tools. NICAD cannot detect type-4 semantic clones

Chanchal K. Roy and James R. Cordy [8] examines the effectiveness of a new language. This method accurately finds near-miss clones using an efficient text line comparison technique. Using Agile parsing it provides user specified flexible pretty-printing to remove noise, standardize formatting and break program statements into parts such that potential changes can be detected as simple line text differences, it provides extraction of potential clones, using transformation rules it provides flexible code normalization. It is to about the finding functional clones in C code.

III. PROBLEM FORMULATION

The objective is to find the duplicated components in the software. Due to the detection of the clones in the software may reduce the software metrics such as to reduce the maintenance cost and also to reduce the number of lines of code, Cyclomatic complexity and coupling in the software programs or software code.

The work involves identification of cloned components and also the prediction of clone evolution content in an open source software applications. The identification of duplicated and nearly duplicated code is also immensely helpful in the field of bug prediction. It is also useful for reducing the Corrective Maintenance and Preventive Maintenance which involves modification of code content to solve and prevent problems in the software respectively. Because if we can identify and detect the cloned areas, the defect in all the similar code fragments can be resolved at once.

The software clone evolution prediction is immensely helpful in Perfective Maintenance and Adaptive Maintenance because the effort required to evolve a software is also dependent on the amounts of cloned contents in the software. Clone evolution prediction is also helpful in the validation of many software evolution hypotheses. The customer also can evaluate the evolution of clone content for taking decisions on purchasing new versions of software applications.

IV. FUTURE SCOPE AND CONCLUSION

The detailed analysis of the relationship of software metrics and software clones. It can give a clearer picture of clones in software. And can also model the increasing and decreasing temporal patterns of the software clone evolution using advanced modelling techniques. This can also predict whether code-refactoring is required in case the code fragments smell bad i.e. the error-prone code fragments in the software. In future this approach can be integrated with other approaches like abstract syntax tree based approach and the program dependence graph approach to make this a hybrid approach to efficiently detect semantic clones.

The technique that detects clones (type-1 and type-2) by metrics based approach for filtering code and after that it uses token based comparisons to detect code clone. The technique detects clones by other algorithm to detect whether two clones really are clones of each other and it is also able to detect the type 3 clone near miss clone by using hash algorithm. The technique can also detect code plagiarism in student's computer lab programs.

REFERENCES

[1] Deepali, Ankur Gupta, Chirag Batra "Hybrid approach for Detecting Code Clone by Metric and Token based comparison," Volume 7, No. 6(Special Issue), November 2016, 978-93-85670-72-5 © 2016 (RTCSIT) pp. 297-302,2016.

[2] JAYADEEP PATI, BABLOO KUMAR, DEVESH MANJHI, AND K K SHUKLA "A Comparison Among ARIMA, BP-NN, and MOGA-NN for Software Clone Evolution Prediction," 2169-3536, 2017 IEEE, VOLUME 5, 2017, pp.11841-11851,2017
[3] Stefan Bellon, Rainer Koschke," Comparison and Evaluation of Clone Detection Tools," IEEE Transactions on Software Engineering, Vol. 33, No. 9, SEPTEMBER 2007,pp.577-591,2007
[4] Elizabeth Burd, John Bailey," Evaluating Clone Detection Tools for Use during Preventative Maintenance," Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'02) 0-7695-1793-5/02 $17.00 © 2002 IEEE
[5] Shruti Jadon," Code Clones Detection Using Machine Learning Technique: Support Vector Machine," ISBN: 978-1-5090-1666-2/16/$31.00 ©2016 IEEE, pp.299-303
[6] Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant'Anna, Lorraine Bier," Clone Detection Using Abstract Syntax Trees," Copyright 1998 IEEE. Published in the Proceedings of ICSM'98, November 16-19, 1998, pp.1-10, 1998
[7] Chanchal K. Roy ," Detection and Analysis of Near-Miss Software Clones," 978-1-4244-4828-9/09/$25.00 2009 IEEE Proc. ICSM 2009, Edmonton, Canada ,pp.447-450
[8] Chanchal K. Roy and James R. Cordy," NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization," The 16th IEEE International Conference on Program Comprehension, 978-0-7695-3176-2/08 $25.00 © 2008 IEEE DOI 10.1109/ICPC.2008.41
[9] Jeffrey Svajlenko Chanchal K. Roy," Evaluating Clone Detection Tools with Big Clone Bench." 978-1-4673-7532-0/15/$31.00, 2015 IEEE, ICSME 2015, Bremen, Germany, pp.131-140
[10] Jaweria Kanwal, Katsuro Inoue , Onaiza Maqbool," Refactoring Patterns Study in Code Clones during Software Evolution," 978-1-5090-6595-0/17/$31.00 ,2017 IEEE, pp.45,46
[11] Ripon K. Saha, Chanchal K. Roy, Kevin A. Schneider, Dewayne E. Perry," Understanding the Evolution of Type-3 Clones: An Exploratory Study," 978-1-4673-2936-1/13, 2013 IEEE, pp.139-148
[12] Richard Wettel Radu Marinescu," Archeology of Code Duplication: Recovering Duplication Chains From Small Duplication Fragments," 0-7695-2453-2/05 $20.00 © 2005 IEEE
[13] Toshihiro Kamiya," CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 7, JULY 2002, pp.654-670
[14] Yang Yuan, Yao Guo, "CMCD: Count Matrix Based Code Clone Detection," apsec, pp.250-257, 2011 18th Asia-Pacific Software Engineering Conference, 2011.
[15] Gehan M. K. Selim, King Chun Foo, Yung Zou," Enhancing Source-Based Clone Detection Using Intermediate Representation," 2010 17th Working Conference on Reverse Engineering, 1095-1350/10 $26.00 © 2010 IEEE DOI 10.1109/WCRE.2010.33,pp.227-236
[16] Brenda S. Baker," On Finding Duplication and Near-Duplicate ion in Large Software Systems," 0-8186-7111-4/95 $4.00 0 1995 IEEE,pp.86-95
[17] Flavius-Mihai Lazar, Ovidiu Banias," Clone detection algorithm based on the Abstract Syntax Tree approach," 978-1-4799-4694-5/14/$31.00 ©2014 IEEE, pp.73-78

Authors Profile

Ms. V. Guna pursed her Bachelor Degree in Computer Science And Engineering from Jawaharlal Nehru Technologiucal University, Ananthapurumu in 2015 and doing masters Degree in Computer Science from Sree Vidyanikethan Engineering College, Tirupathi, Affiliated to Jawaharlal Nehru Technological University Ananthapuramu in 2016-2018 Project Trainee in Sree Vidyanikethan Engineering College, Tirupathi.

Dr. M Sunil Kumar has completed B.Tech in Computer Science & Information Technology from JNT University, M.Tech in Computer Science from JNT University.and Ph.D in Computer science and Engineering from SV University Tirupati. Presently he is currently working as Professor & head in the Department of CSE, Sree Vidyanikethan Engineering College, A.Rangampet, Tirupati, A.P. His main research interest includes Software Engineering, Software Architecture, Information Retrieval,Database Management Systems and optimization techniques.

