International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-8, Issue- 9S2, July 2019

An Efficient Type 4 Clone Detection Technique for Software Testing

Shikha Golyan, Rohit Aggarwal

Abstract: Software testing is a procedure which is utilized to distinguish the bugs and reveal it. Software testing is a procedure and control moreover. It is not quite the same as programming improvement. It ought to be viewed as that is a piece of programming improvement. Clone testing is one of the sorts of testing. It is utilized to check the guile in the product. While build up any product for sparing time and exertion, programming designer. Reorder program code over and over. So if any bug found in one module is duplicated in each duplicate. There are numerous duplicates of code present and no record of such duplicates is available. This will make hard to fix such bugs and upkeep of existing programming. Code clone is one of the components making programming upkeep increasingly troublesome. There are various kinds of clones present, Type 1, Type 2, Type 3, Type 4. The current calculation has identified clone in Type 3 as it were. In the proposed work we will improve algorithm which will probably identify clone in TYPE 4 too. This expands the presentation of the framework.

Keyword: Algorithm, code clone, clone testing, software testing

I. INTRODUCTION

Programming code cloning is widely utilized by originators to cause code in which they have sureness and which reduces movement costs and improves the thing quality. Programming clone examination in the prior years was generally focused on the affirmation and examination of code clones, while examination recently stretches out to the entire extent of clone association. Reusing programming through facsimileing and staying is a steady torment in programming improvement paying little regard to the manner in which that it causes true help issues. The path toward duplicating a code is known as code clone. A couple of programming designers perform code cloning purposely or coincidentally during the improvement of use or programming. It has been mulled over that 30% of the code in most of the item associations is copied code.

So it is major to understand that why the code has been replicated, why there is a need to duplicate the code, how the copied or cloned code contrarily influences the upkeep and progression. For upkeep and progression reason, a couple of stages like clone recognizable proof, examination and backing have transformed into an important zone of research for certain researchers. Regardless of the way that cloning has various inclinations in programming organizations.

It saves the product architect's time, reusability of code is basic for a novice in the business. In any case, when we reuse the code, the overhead similarly augments. So cloning has a blurred side as well. The huge matter of concern is the upkeep of the made programming. Once in a while the cost for help outperforms more than the cost of improvement.[1]

A. Software Cloning

Clones are the regions of source code which are significantly practically identical; these regions of closeness are called clones, clone classes, or clone sets. While there are a couple of reasons why two territories of code may be practically identical, the majority of the clone examination composing credits cloning development to the intentional recreating and duplication of code by engineers; clones may in like manner be attributable to normally made code, or the objectives constrained by the usage of a particular structure or library. Despite these, some various issues, including programming architect's direct, for instance, torpidity and the inclination to repeat typical game plans, advancement requirements, code get limit and outside business forces have impacts on code cloning. Cloning is the unnecessary duplication of data whether it is at design level or at coding level. Cloning works to the detriment of extending lines of code without adding to when all is said in done productivity. Same programming bugs and distortions are reproduced that reoccurs all through the item at its progressing likewise its help arrange. It results to inordinate help costs moreover. So cut paste programming kind of programming reuse deceivingly raise the amount of lines of code without expected diminishing in upkeep costs related with various sorts of reuse. Along these lines, clones, is a promising technique to reduce the upkeep cost in future.[2]

Reasons behind copying a code by developers are:

- Code part can be easily duplicated than making the code with no arrangement. In expansion, the part may as of now be tried so presentation of a bug appears to be more uncertain.
- Evaluating the exhibition of a software engineer by the measure of code the individual in question produces gives a regular motivator for duplicating code.
- Efficiency examinations may make the cost of a procedure call or system bring seems, by all accounts, to be too high an expense. In mechanical programming headway settings, time weight together with first and second aides lead toward a ton of chances for code duplication.

Revised Manuscript Received on July 18, 2019.

Shikha Golyan, Student, Masters of Technology in Computer Science and Engineering from Meerut Institute of Engineering and Technology, Meerut, U.P, India.

Rohit Aggarwal, M.Tech – Computer Science & Engineering, Meerut Institute of Engineering and Technology, Meerut. U.P, India.

B. Types of Code Clones

Types of code clones: Mainly four sorts of code clones are accessible which are clarified underneath:

Type 1: This type comprises similar code clones. These code clones permit alteration in white space and comments only.

Type 2: In this type code clones are similar copies in semantic and syntactic manner.

Type 3: In this type code clones are copied remains with additional adaptations by means of variation, addition or elimination of statements.

Type 4: In this type code clones rely on function resemblance but differ in syntax.

II. CLONE TESTING

Software engineering is a methodology related with advancement, designing procedure and continuance of programming. Therefore, quality of product, detection of bugs and prevention of bugs from system through testing or scrutiny are the major objectives of software engineering. During the advancement of any product so as to spare time and exertion, programming designers reorder program code more than once. In this manner any bug recognized in one module is recreated in each duplicate. Various duplicates of code are available however no record of such duplicates exists. This makes troublesome the fixing of bugs and safeguarding of present programming. Code clone is one of the dynamic which makes programming upkeep further complex [3].

The examination of programming applications shows that clones produce by collecting couple of extra procedure which are practically equivalent to yet not like some open rationale inside a framework. A code clone is only an alike or copy code in a source code or produced either through reenactment or a few adjustments. These cloned codes summed up with high protection consumption of programming and turns into the reason of code swelling too. This is because of the reality change in one clone and execution of comparative follow up on regarded clone builds the support cost. These clones may improve the danger of flaws in framework too by expanding the danger of executing conflicting adjustments in the code. Prior looks into have shown that around 7% to 23% of the source code in a programming framework contains code clone. Different apparatuses are accessible for the discovery of code clones, yet these instruments are not viable in the evacuation of these clones. These code clones require the best possible working of programming. Consequently, the possibility of refactoring or particularity can be connected for improving the reusability and practicality of programming from clone code.

A. Clone Detection Techniques

Clone detection techniques are mainly categorized into four major categories. These techniques are described below:

- Textual approach: Textual techniques require less standardization or manipulation of code. In this technique, mainly line by line examination is performed. This correlation depends on two sorts of coordinating for example basic line coordinating and parameterized line coordinating. This methodology depends on string primarily.
- Lexical approach: In this approach, source code is changed over into tokens with the assistance of lexical principles. After this, an examination between these tokens is performed [3].
- Syntactic approach: A theoretical tree is produced in this technique. Source code is changed into parse tree with the help of parse code. At that point hypothetical tree is prepared either through tree coordinating or metric for the disclosure of clones.
- Semantic approach: Source code is characterized in the form of program dependence graph in this approach. Statements and terminologies are demonstrated by nodes. Edge represents control and information dependence.

III. LITERATURE REVIEW

Brent van Bladel and Serge Demeyer (2019) proposed a new approach for detecting a semantic code clones in test code [4]. For this motive, symbolic execution was utilized for the generation of test behavior demonstration. These test behaviors were compared later. The proposed approach utilized Apache Commons Math Library's test set. The tested outcomes depicted that the proposed technique discovered 755 clone pairs with an accuracy of 98%. It was also recognized that 259 pairs out of 755 discovered clone pairs were the type 4 clones. This was established that proposed approach was both reasonable and valuable for the inspection of semantic clones in test code.

Swati Sharma and Priyanka Mehta (2016) presented a novel algorithm for detecting function clones within software system. Function clones increased the handling expenses so they were considered dangerous for the software system [5]. The main objective of proposed algorithm was the handling of the type-IV clone. Merely type-IV clone could identify the function clones within source code. The conclusions showed the comparison of proposed and existing approach as well. The comparative outcomes demonstrated that modification in existing approach increased effectiveness; decreased handling charges and detected both function clones and code clones. In this study, the modified algorithm was projected for the detection of function clones in source code. In future, supplementary improvement approaches will discover the clone in sequence. These approaches will provide information about the existence of clones in lines. This approach will provide sternness to the discovered clones. The sternness of the clones will give enhance scrutiny by means of code clone discovery.

Rowyda Mohammad, AbdEl-Aziz, Amal Elsayed, Aboutabl, Mostafa-Sami Mostafa, (2012) proposed a novel approach for the extraction of precise clones from object slanted source code with the help of Differential File Comparison Algorithm [6]. The main aim of this method was the detection of cloned code. These cloned codes were the reason of crosscutting issues which reduced the system reincarnate and sustainability. Differential File Comparison

Algorithm was used to detect the dissimilar lines of codeamong two source codes files. The remaining lines of code were similar and identified as clones. Then that clones were taken out from code. In this process, mainly three phases were involved i.e. Source code normalization, Differential File Comparison, Extracting Exact Clones. The initial phase required only small alteration of code. White spaces and comments were detached from code during alteration. Differential File Comparison Algorithm was used to find out the dissimilarity of lines among two files in the second phase. The untransformed lines were recognized as clones. The issue of highest frequent subsequence was resolved through the maximization of line sizes which were not changed. Precise clones from two specified source codes were extracted in the third phase. Identical lines were identified as clones which resulted in the extraction of clones from two source files. But, this technique was applied on C# language. This approach was employed on just two source files. Though this approach was easy and utilized small time but it was able to detect type 1 clones merely. In future, this technique can be modified for detecting type 2 or type 3 clones and applies on more than two source codes.

Mohd. Ehmer Khan (2010) presented a summary of different software testing approaches [7]. The major objective of testing was the detection of faults with the help of different techniques. The main issue was the discovery of an appropriate approach for the elimination of errors. In this study, dissimilar kinds of testing techniques were described. Accuracy testing was applied for the identification of correct performance of the system and united the white box and black box testing characteristics. The explanation of plan, strategy, execution and reporting was provided through performance testing. The performance testing was further divided into load and stress testing. Reliability testing was executed for the removal of bugs prior to the release of the product and provided a explanation of security testing.

Salwa K.Abd-El-Hafiz (2013) proposed a metric based data mining technique for the discovery of software clone [8]. The proposed approach was identified as Fractal Clustering Algorithm. This data mining based approach was used to detect the clone clusters rather than couples. The proposed approach involved four stages. In the initial stage, a source file was provided as input for preprocessing. In second stage, scrutinized fragments and allied metrics were retrieved. In third stage, fractal clustering was used for the division of fragments suites into little clusters. The proposed approach placed functions into three dissimilar clusters i.e. primary, intermediate, single clusters. Type 1 and type 2 clones were clustered in the primary cluster. The metric values of these clones were similar and showed similar level of resemblance linked with selected metric suite. Type 3 clones were clustered in intermediate cluster. These clones had identical metric value and came under some threshold value. Single and no-similar clones were clustered in the single cluster. This procedure was performed with the help of eight metrics for the detection of every function scheme. These eight metrics were deployed into four dissimilar metrics set {m1,m2…..mD}. Big sized (SNNS), average sized (Weltab) and open source C program were used for the evaluation of this approach. The approach was not able to detect type 4 clones and this was the main drawback of this approach. The outcomes of proposed approach varied with the variation of threshold values.

IV. PROPOSED WORK

Code cloning is characterized as the strategy of duplication and adjustment of code or the advancement of duplication of code parts in the source code. Clone bunch commonly comprises code clones those are clones to each another. Clone disclosure is a technique utilized for the revelation of functions which are exact duplicates of another functions in the programming framework. Type 1 clones can be detected easily while type 4 clones are difficult to detect.Type 4 Cloning is utilized to discover clones in a specific function. However, the detection of type 4 clone is complex because it makes clone of all copied functions rather than of merely copied lines. This issue is resolved using pattern matching algorithm. In this algorithm merely matched values are cloned and other values are not cloned.

A. Scope of Study

This research work has a wider range. The handling of type 4 cloning is difficult since it clones functional data as well. This type of cloning can be used for some other aims also by eliminating this fault. Difficulty of function can be eliminated through the combination of type 4 cloning and pattern matching. The complete function is cloned in type 4 cloning. However in pattern matching algorithm, just couple of lines are cloned. This procedure is a less time taken, it is found that it is faster also when compared to accessible function.

B. Objectives

Major objectives of this research are as follows:

- To discover the Type 4 clone.
- To eliminate the issue of complete function cloned of type 4 with the help of pattern matching algorithm.
- To create prototypes for reducing the cloning of all functions.
- To reduce the difficulty of functions.
- To implement time saving procedure.

C. Research Methodology

This approach can be used for several purposes after eliminating this error. Difficulty of function can be eliminated through the combination of type 4 cloning and pattern matching. The complete function is cloned in type 4 cloning. However in pattern matching algorithm, just couple of lines are cloned. This is a less time taken procedure and faster as compared to accessible function. The category of code cloning is a displayer of difficulty and the extent of complexity in the detection and identification of clone. In this circumstance, type 1 clone can be identified most effectively and the location of sort 4 clone is generally troublesome. Type 4 Cloning is used to discover clones in a specific function. However, the detection of type 4 clone is complex because it makes clone of all copied functions rather than of merely copied lines. This issue is resolved using pattern matching algorithm. In this algorithm merely

matched values are cloned and other values are not cloned. In this study, improvements are made in the accessible algorithms. The clones can be detected from the entire code in accessible algorithm. In modified algorithm, the clone code under particular function is verified as well. In such a situation, when function under which verification is being performed comprises the similar name of cloned code, and then entire code is identified as the cloned code. If the function names are different, then the cloned code discovery procedure persists till the detection of complete code. The tested outcomes demonstrated that the modified algorithm is quite competent and discovers the cloned code more proficiently in minimum span of time.

V. RESULTS

The interface is generated for clone testing as demonstrated by the figure 1. The tool analyzes the effectiveness of both accessible and proposed algorithm.

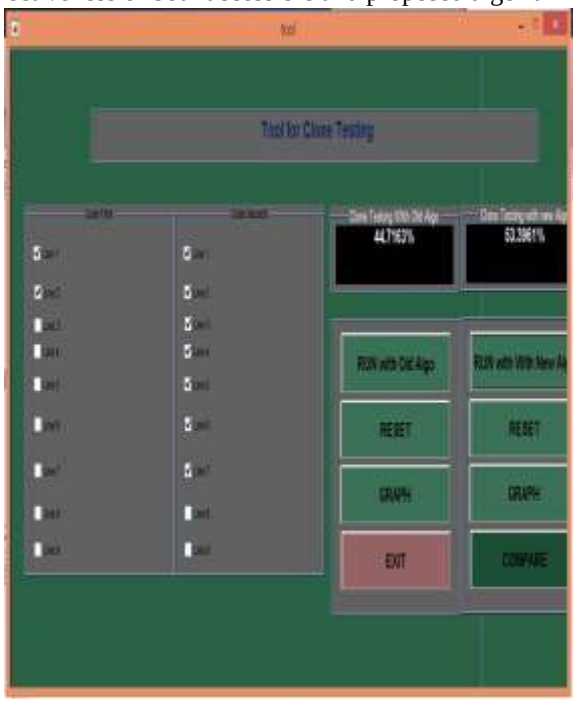

Fig1. Comparison of efficiency between old and new algorithm

Figure 1 shows that when marked lines(line 1 and line 2) of first code is compared with marked lines(line 1 to line 7) of second code are examined , the tool shows that 44.71 % code is cloned with old algorithm and 53.39% code is cloned with new algorithm. Figure 2 depicts the graphical representation of efficiency with new and old algorithm which clearly indicates that efficiency is increasing with new algorithm.

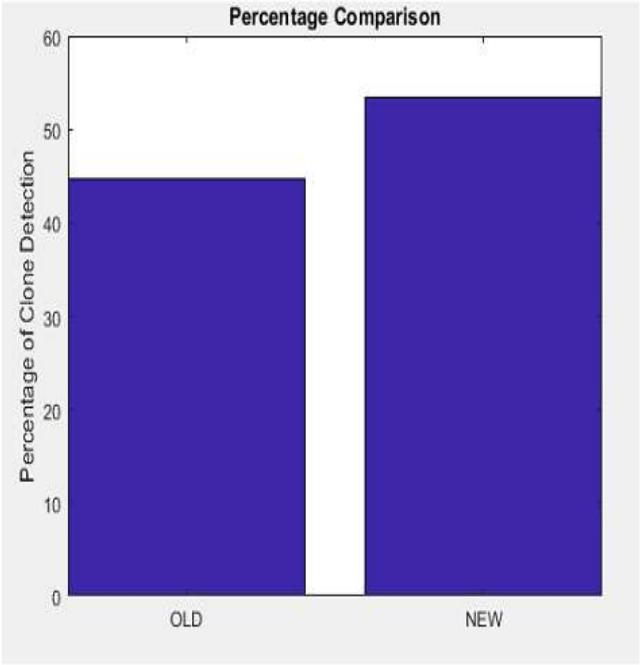

Fig.2: Graphical Comparison of efficiency with old and new algorithm

VI. CONCLUSION

By eliminating out this slip-up we can use this for different purposes. When we join type 4 with pattern matching algorithm, complexity of the function is emptied. In sort 4 whole functions are cloned anyway with pattern matching algorithm few lines are cloned which is proficient methodology and snappy than the present capacity. The kind of code clone is a pointer of multifaceted nature similarly as the component of inconvenience in perceiving and recognizing the clone. For this circumstance, type 1 is the least requesting to perceive and type 4 the most problematic. Type 4 cloning is used for the identification of clones in a particular function. In any case, type 4 is difficult to in light of the fact that it make clone of all the limit it is imitated rather than simply the copied lines. To overcome this issue we have to use the pattern matching algorithm with the objective that only matched values are cloned. In future another algorithm will be proposed which can perceive type 1 type 2 clone in addition. The results furthermore exhibit the connection of new algorithm with old algorithm. Through correlation, we look at that redesigning old algorithm increase efficiency, decrease upkeep cost , discover function clones as well as code clones.

REFERENCES

1. Sandeep Kaur, Geetika Chatley and Bhavneesh Sohaal, "Software Clone Detection:A Review", 2016, International Journal of Control Theory and Applications

2. Swati Sharma, Priyanka Mehta, "A Literature Survey on Software Clone Testing", International Journal of Science, Engineering and Technology Research (IJSETR) Volume 5, Issue 4, April 2016
3. Manpreet Kaur and Rupinder Singh, "A Review of Software Testing Techniques", 2014, International Journal of Electronic and Electrical Engineering
4. Brent van Bladel, Serge Demeyer, "A Novel Approach for Detecting Type-IV Clones in Test Code", 2019 IEEE 13th International Workshop on Software Clones (IWSC)
5. Swati Sharma, Priyanka Mehta, "To Enhance Type 4 Clone Detection in Clone Testing", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (2), 2016, 967-971
6. Rowyda Mohammad, AbdEl-Aziz, Amal Elsayed, Aboutabl, Mostafa-Sami Mostafa, "Clone Detection Using DIFF Algorithm for Aspect Mining", International Journal of Advanced Computer Science and Applications, Vol. 3, No.8, 2012
7. Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors", IJCSI, International Journal of Computer Science, Issues, 7, 2010
8. Salwa K.Abd-El-Hafiz, "Code Cloning: The Analysis, Detection and Removal", International Journal of Computer Applications, Volume 20- No.7, April 2013
9. C. K. Roy, "Detection and Analysis of Near-Miss Software Clones",Ph.D. Thesis, Queen'sSchool of Computing, Queens University, 2009-08-31.
10. Jovanovic Irena, "Software testing methods and techniques", 2002.

AUTHORS PROFILE

Shikha Golyan pursuing Masters of Technology in Computer Science and Engineering from Meerut Institute of Engineering and Technology, Meerut. Area of Specialization: Software Engineering

Mr. Rohit Aggarwal, M.Tech – Computer Science & Engineering. Area of Specialization: Software Testing, Neural Network & Fuzzy Logic, & Data Sciences.