IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 4, Ver. II (Jul.-Aug. 2016), PP 15-19
www.iosrjournals.org

Comprehensive Study of Software Clone Detection Techniques

Sunayna[1], Kamna Solanki[2], Sandeep Dalal[3], Sudhir[4]
[1, 4] Research Scholar, M.D.University, Rohtak
[2, 3]Assistant Professor, M.D.University, Rohtak

Abstract: *Software cloning means duplication of source code. It is most basic means of software reuse. A software clone is a code fragment which is identical to another in the source code. Clones are harmful for software maintenance because it increases the complexity of system and maintenance cost. If we detect software clones it can decrease software maintenance cost. Many code clone detection techniques have been proposed for this purpose. Several studies show that about 5% to 20% of software system can contain duplicated code which is results of copying existing code fragments and around 60% of the efforts of an organization are wasted in maintaining this. The main disadvantage of code duplication is that if a bug is detected in a code fragment; all the other fragments similar to it should be checked for the possible existence of the same bug. By using different clone detection techniques, we can detect code clones which increase the efficiency of software maintenance process and thus decreases the maintenance cost.*

Keywords: *Software cloning, Types of software clone, Software clone detection techniques, Comparison of different clone detection techniques.*

I. Introduction

Code clones are the sections of very similar or identical code. These are obtained by reusing of code fragments by copying and pasting with or without any major or minor modifications [1]. Duplication of code occurs generally during the development of large software system. Code cloning is a form of software reuse and exists in every software project. There is no proper definition of code cloning. In many cases, one cannot be sure that fragment of code is copied from another source. Many terms were used for code cloning by different researchers like: Baxter et al. [2] stated that a clone is "*a program fragment that is identical to another fragment"*. Krinke [3] uses the term "*similar code*". Ducasse et al. [4] used the term "*duplicated code*", Komondoor and Horwitz [5] also defined the term "*duplicated code*" and use "clone" as an instance of duplicated code.

There are a number of reasons for cloning of source code. Main reason is that programmer find it is cheaper and quicker to use the copy and paste feature than writing the code. Sometimes programmer try to implement new functionality but find some working code that performs identical computation to the desired. In this case the programmer will copy it entirely [6]. While it is good reuse practice it complicates the maintenance process. Code cloning is a serious problem in software industry. Code cloning looks easy and cheap during software development phase but it makes software maintenance more difficult. Software clone has number of negative effects on software quality, increases the amount of code, maintenance cost and bug probability.

Cloning creates problem in software maintenance due to following reasons:-

1. Cloning unnecessarily increases program size that leads to increase in maintenance efforts.
2. If we make changes in one clone like bug fixing it need to be made to other clones as well again increasing maintenance efforts.
3. If changes to duplicated source code are made inconsistently, it can introduce new bugs.
4. Due to code cloning specific purpose of each section remains hidden.
5. Increases the maintenance costs.

Due to the rapid increase in the code clones and the resulting maintenance problems, more and more focus is being shifted to the detection of the various types of code clone and many techniques for clone detection are introduced.

Types of code clone:-

There are four types of code clone which are described below.

1. Type 1: It is also called exact copy clone. In this type some variations exist in change of comments or in white spaces.
2. Type 2: It is syntactical same copy. In this type of code clone literals are changed e.g. name of variables and name of functions are changed. It is difficult to detect from type 1.
3. Type 3: It is code clones in which lines are added or deleted or interchanged. It is also syntactic clone.
4. Type 4: It is a code clone which is not created intentionally. Developer is un-known about the presence of similar code. This type of code clone is very difficult to detect.

DOI: 10.9790/0661-1804021519

II. Literature Review

Recently, code clones have received much attention. Code clones are identical or similar code fragments to one another in source code.

Software cloning and its detection techniques is an important research area in IT industry. Out of the various factors, one factor that results in software piracy is code clones that are becoming increasingly critical to the IT world. For customers and IT company code clones cause a serious threat to the security, quality and maintenance. Therefore, it is very necessary to detect and check the various types of code clones. Many researchers have worked in this field.

Davey et al. (1995) used the concept of neural network with metrics to detect code clones. In this approach features of code blocks are stored in vectors and then trained in Dynamic Competitive Learning (DCL) Model to find similarity. Type-1, Type-2, Type- 3 clones can be detected by using this approach [7].

Kontogiannis et al. (1996) determined various factors like number of functions called (fanout), ratio of input/output variables to the fanout, McCabe cyclomatic complexity, Modified Albrecht's function point metric etc. to determine whether the code fragments are clones or not [8].

Baxter et al. (1998) defined code clones as the segments of code that are similar according to some definition of similarity. While they provide a threshold-based definition of tree similarity for near-miss clones ie Type-3 clones [9].

Ducasse et al. (1999) proposed method instances of line-based clone detectors. In these methods, every line of code is compared after white space and tabs are removed. These methods are language-independent because they compare lines of code textually and have more efficiency [4].

Burd and Bailey (2002) conducted one of the first experiments for comparing clone detection tools. They compared three state-of-the-art clone detection and two plagiarism detection tools. They began by validating all the clone candidates of the subject application obtained with all the techniques of their experiment to form a human oracle, which was then used to compare the different techniques in terms of several metrics to measure various aspects of the reported clones [10].

Fabio et al. (2004) designed the tool Datrix for extending the quality of Java code. The metrics calculated by this tool are useful for detecting clones in the Java software and it is easy to use too. Metrics are calculated from names, layout expressions and control flow of functions. Metrics-based approaches have also been applied to finding duplicate web pages and clones in web applications [11].

Al-Ekram et al. (2005) have conducted a promising empirical study on cloning focusing on C/C++ systems from two different domains. They examined different clone types (e.g., accidental clones) by analyzing clones across systems in the same domain [12].

Balint et al.(2006) correlate code clones with time of modification and with the developer that modified it for detecting the patterns of how developers copy. Based on these patterns they develop a visualization tool called Clone Evolution View to represent the evolution of code clones [13].

Koschke et al. (2006) and Jiang et al. (2007) presented tree-based approaches. In Koschke et al.'s method, ASTs are compared with a suffix tree algorithm to increase the detection speed. On the other hand, Jiang et al. use a locality sensitive hashing algorithm to detect code clones. With this algorithm, Jiang et al.'s method can detect Type-3 clone [14,15].

Cordy et al. (2009) stated that there are two main kinds of similarity between code fragments. Fragments can be similar based on the similarity of their program text, which is called textual similarity or they can be similar based on their functionality which is called functional similarity. There are four clone types in total, in which the first three are textual and the last one is functional [16].

Hummel et al. (2010) proposed an index based code clone detection methodology. Their method firstly replaces user defined identifiers with special tokens in every line of the source code, then hash values are calculated from them. Next, the method stores their hash values, their line numbers, and their files names into the database. By using the database, lines that are duplicated with specified lines can be instantly obtained [1].

Yoshiki Higo et al. (2011) discussed PDG approach of code clone detection. They developed a prototype tool, and applied it against open source software. The experiment showed that the proposed method could obtain code clones within a short time period and its detection result was quite similar to the detection result of an existing PDG-based detection tool [17].

Nguyen H.A et al. (2012) presented a tool JSync for clone management system. This tool provides support to clone detection and updating, clone change management, clone synchronizing, clone consistency, validating, and clone merging. It represents abstract syntax tree which measures for code similarity [18].

Girija Gupta et al. (2013) introduced work metric based methodology which is utilized to distinguish the potential clones and after that upgrading of code is done. This methodology has capacity to detect semantic clones [19].

Patil et al. (2014) detected code clones using Decentralized Architecture and Parallel Processing. CFG and weighted graph were used by them for finding all four types of clone. The technique can achieve optimization in the reference of time and space by implementing a decentralized architecture. Decentralized architecture can improve flexibility of the tool and parallel programming can reduce the detection time [20].

Shahid et al. (2015) analyzed various clone detection tools. The study would help to decide which tool is best suitable for detection of code clones. They present the background concepts of cloning, a generic clone detection process and a comparison of clone detection tools [21].

Clone Detection is of great concern for better maintenance and quality of software system. Systems containing code clones are highly susceptible to bugs and become hurdle for better evolution of software system. So, it is an large area of research from many years and results into various clone detection techniques and tools based on them. But certain limitations are associated with each clone detection technique and tool. Following problems are identified in existing work:

i. A novel approach is required to detect code clones efficiently.
ii. Both syntactic as well semantic clones should be detected by any tool.
iii. As the clone detection process has many phases, so tool should be automated and light-weighted so that each phase is executed without any computational resources.
iv. Many false positive clones are detected by tools which should be removed to get high precision value.

We will study and compare various clone detection techniques on the basis of some parameters in order to pick the right one for a particular purpose of interest.

III. Various Techniques To Detect Software Cloning

The various techniques for detection software cloning are:

- ***Text-based Cloning Techniques:-***

There are various clone detection techniques that are based on text-based methods. In this approach, the target source code is considered as sequence of lines or strings. Two code fragments are compared with each other to find sequences of same strings. Once two or more code fragments are found to be similar they are returned as clone pair or clone class by the detection technique. Because of the purely text-based approach, detected clones do not correspond to structural elements of the language. This approach can detect Type-1 code clone but cannot detect the structural type of clones. There are several problems that can arise in a text-based detection technique. Some of these are as follows:

1. Line Break:- Code portions with line break relocation are not detected as clones and if detected then as shorter clones.
2. Identifier changes:- Changes of identifier names may not be handled in text-based method which is not efficient.
3. Parenthesis removal or adding a single statement:- For example, a single statement can be with or without surrounded by begin-end brackets (e.g., *a*\ and *b*\) just after if, else or for statements. In text-based detection technique, the presence of *a*\ and *b*\ pair in one code segment but not in the other creates a big problem while comparing the two code fragments and may detect as distinct fragments even if they are exact copy clones. Therefore,different kinds of coding style can create problems in this method.

- ***Token Based Cloning Techniques:-***

Token based approach is also known as lexical approach. This approach uses parser or lexer for the transformation of source code into a sequence of tokens. In the token-based detection approach, the entire source system is lexed or parsed to a sequence of tokens. This sequence is then scanned for finding duplicated subsequences of tokens and finally, the original code portions representing the duplicated subsequences will be returned as clones. This approach though more efficient than text based approach over minor code fragments if there exists blank spaces and comments but its accuracy level is not satisfactory because false clones will be introduced in the code while conversion of source code in the token sequence [6]. Compared to text-based approaches, a token-based approach is usually more robust.

One of the leading token-based techniques is *CCFinder* [17] .In this approach first, each line of source files is divided into tokens by a lexer and the tokens of all source files are then concatenated into a single token sequence. The token sequence is then transformed, i.e., tokens are added, removed, or changed based on the transformation rules. After that each identifier related to types, variables, and constants is replaced with a special token. This identifier replacement makes code fragments with different variable names clone pairs. A su±x-tree based sub-string matching algorithm is then used to find the similar sub-sequences on the transformed token sequence where the similar sub-sequence pairs are returned as clone pairs or clone classes.

- ***Tree-based Cloning Techniques:-***

In the tree-based approach a program is pared to a parse tree or an abstract syntax tree (AST) with a parser or lexer. Similar subtrees are then searched in the tree with some tree matching techniques and the corresponding source code of the similar subtrees is returned as clone pairs or clone classes. The AST contains the complete information about the source code.

One of the leading AST-based clone techniques is that of Baxter et al.'s *CloneDR*. [2]. In this approach, a compiler generator is used to generate an annotated parse tree ie AST and compares its sub-trees by characterization metric through tree matching technique. Source code of similar subtrees is then returned as clones.

The level of accuracy is considered good in this approach but it results in unstable scalability because it depends on the algorithm that is being used to build and compare the trees [14].

- ***Program Dependency Graph Based Cloning Techniques*:-**

Program Dependency Graph (PDG)-based approaches [3,5] go one step further in obtaining a source code representation of high abstraction than other approaches because it considers the semantic information of the source. PDG contains the control flow and data flow information of a program and hence carries semantic information. Once a set of PDGs are obtained, isomorphic subgraph matching algorithm is applied for finding similar sub-graphs which are returned as clones. The dependency graphs needed to be built for this approach and the accuracy of these graphs have to be good, as a simple form of error in the dependency graph can lead to the building of codes which makes it difficult to detect the clones. PDG- based detection approach is more effective than other techniques as it can detect non- contiguous code clones but it is a costly process to obtain.

- ***Metric Based Cloning Techniques:-***

In this approach metrics are used to measure clones in software after the calculation from source code. This approach parses the source code to its AST or PDG representation for the calculation of metrics [9]. For the purpose of calculation of metrics from source code various tools like Columbus, Source monitor are available [6]. This approach provides high accuracy and scalability level. Metric-based approach has been also applied for finding duplicated web pages or clones in web documents. One limitation is that this approach does not detect copy-paste at other granularity such as segment-based copy-paste, which occurs more frequently than function-based copy-paste.

IV. Comparison of Different Clone Detection Techniques

The parameters with which the clone detection techniques can be compared are known as clone detection challenges. Some of the parameters used for comparing the different techniques are as follows:-
Portability: The detection technique should be portable in terms of multiple dialects and languages. A clone detection technique must be portable and easily configurable for different types of dialects and languages.
Precision (Accuracy): The technique should be good enough so that it can detect less number of false positives i.e., the tool using that technique should find duplicated code with higher precision.
Recall: The tool using a particular technique should be capable of locating and finding most or even all of the clones of a system. .
Scalability: As duplication is the most problematic in complex systems, the tool using the technique should be able to find clones from large code bases with efficient use of memory.
Robustness: A good tool should be robust in terms of the different editing activities that are applied on the copied code fragment so that it can detect different types of clones with higher precision and recall.

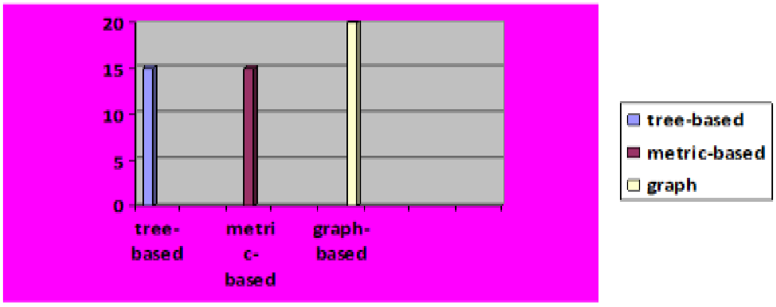
Name	Portability	Accuracy	Robustness	Scalability
Text-based	High	High	Low	Relative to comparison algorithm
Token-based	Medium	Low	Limited	High
Tree-based	Low	High	High	Relative to comparison algorithm
Graph-based	Low	High	Medium	Low
Metric-based	Relative to defined metric.	High	Medium	High

V. Result and Analysis

From the literature review it is concluded that 20% of researchers have studied about cloning and various techniques and tools to detect software clones. They have compared various clone detection tools on the basis of various comparison parameters.

15% of researchers have used tree based cloning technique to detect software clones. Approximately 20% researchers have used the concept of graph based cloning techniques to detect software clones. Nearly 15% researchers have used the metric based cloning techniques and 15% research was done just to find the software cloning and its types.

Now various new techniques have been invented for clone detection like clone detection using decentralized architecture and parallel processing, clone detection using neural network , by using index based methods and many more new techniques.



VI. Conclusion and Future Scope

Code clones presence is being recognized as an emerging cause of concern in software industry. Due to the presence of code clone maintenance of software has become quite difficult. Therefore, identification of code clones becomes extremely necessary in order to avoid the problems caused by them. We have studied about code cloning and various techniques to detect code clone. A comprehensive survey on the area of software clone detection research has been conducted emphasizing on the types of clones used, their detection mechanism and empirical evaluation.

In future, work can be done on some new technique of detecting code clone. The results of this study may serve as a roadmap to potential users of clone detection techniques, to help them in selecting the right tool or technique for their interests. It may also help in identifying remaining research questions and interesting combinations of existing techniques.

References

[1]. L.Jiang, G. Misherghi, S. Glondu , " DECKARD Scalable and Accurate Treebased Detection of Code Clones", *29th International Conference on Software Engineering (ICSE'07),* July 2007.

[2]. I. Baxter, A. Yahin, L. Moura, M. Anna ,"Clone Detectio Using Abstract Syntax Trees," *in Proceedings of 14th International Conference on Software Maintenance (ICSM'98)* Bethesda , Maryland, November 1998.

[3]. J. Krinke,"Identifying Similar Code with Program Dependence Graphs,"*in Proceedings of 8th Working Conference on Reverse Engineering,* Stuttgart, Germany, October 2001.

[4]. S. Ducasse, M. Rieger , S. Demeyer, " A Language Independent Approach for Detecting Duplicated Code,"*In Proceedings of 15th International Conference on Software Maintenance (ICSM'99),* Oxford, England, September 1999.

[5]. R. Komondoor and S. Horwitz. " Using Slicing to Identify Duplication in Source Code".*In Proceedings of the 8th International Symposium on Static Anslysis (SAS'01),* Paris, France, July 2001.

[6]. C. K. Roy, J. R. Cordy,"*A Survey on Software clone Detection Research Techniques*", Queen's School of Computing, September, 2007.

[7]. N. Davey, P. Barson, S. Field, R.Frank, and D. Tansley, "The development of a software clone detector," *In International Journal of Applied Software Technology,* 1995.

[8]. A. Kontogiannis, R. DeMori, E. Merlo, M. Galler and M. Bernstein, "Pattern matching for clone and concept detection," *In Reverse engineering,* Springer US, 1996.

[9]. R. Koschke, R. Falke, P. Frenzel, "Clone Detection Using Abstract Syntax SuffixTrees", *In Proceedings of the 13th Working Conference on Reverse Engineering (WCRE'06),* Benevento, Italy, October 2006.

[10]. E. Burd, J. Bailey, "Evaluating clone detection ", tools for use during preventative maintain *In: Proceedings of the 2nd IEEE International Workshop on Source Code Analysis and Manipulation, SCAM 2002*, 2002.

[11]. F. Calefato, F. lanubile,T. Mallardo, "Function Clone detection in Web Application. A Semi automated Approach," *Journal of Web Engineering,* 2004.

[12]. R. Al-Ekram, C. Kapser and M. Godfrey. "Cloning by Accident: An Empirical Study of Source Code Cloning Across Software Systems". *International Symposium on Empirical Software Engineering (ISESE'05),* Noosa Heads, Australia,November 2005 .

[13]. M. Balint, T. Girba and R. Marinescu. How Developer Copy. *In Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06),* Athens, Greece ,June 2006.

[14]. R. Koschke, R. Falke, and P. Frenzel. Clone detection using abstract syntax suffix trees. *In Proc. of the 13th Working Conference on Reverse Engineering,* , Oct. 2006.

[15]. L. Jiang, G Misherghi, Z. Su, and S.Glondu. Deckard : Scalable and accurate tree-based detection of code clones. *In Proc. of the 29th International Conference on Software Engineering,* May 2007.

[16]. C.K. Roy, J.R. Cordy, R. Koschlke " *Comparison and evaluation of clone detection techniques and tools":* A qualitative approach. Science of Computer Programming 2009.

[17]. T. Kamiya, S. Kusumoto, K. Inoue, "CCFinder: A multilinguistic token-based code clone detection system for large scale source code", *IEEE Transactions on Software Engineering, 2002.*

[18]. M. de Wit, A. Zaidman, A. van Deursen, "Managing Code Clones Using Dynamic Change Tracking and Resolution", *In Proceedings of IEEE Int'l Conference on Software Maintenance*, 2009 .

[19]. G. Gupta, I. Singh, "A Novel Approach towards Code Clone Detection and Redesigning" 2013, IJARCSSE Volume 3, Issue 9, September 2013.

[20]. Patil et al. "International Journal of Advanced Research in Computer Science and Software Engineering " September - 2014.

[21]. Shahid et al. " International Journal of Advance Research in Computer Science and Management Studies "Volume 3, Issue 1, January 2015.