REVIEW ON VARIOUS CODE CLONE DETECTION TECHNIQUES

Amandeep Kaur[1], Harpreet Kaur[2]
Department of Computer Engineering
Punjabi University, Patiala

Abstract: The code clone is defined as copying the original code and paste it either with or without modifications. Code clone detection process also known as the reprocessing of the original code. These clones make the entire code redundant. The code cloning also leads to the new bugs in the program. The code cloning has a major impact on the software industry as it complex the design of the software project and also difficult to improve the system. Source code cloning represents as significant threat to the maintainability of a software system. To handle these problems, various code clone detection techniques are proposed that are quite efficient in detecting the code clones. The techniques mainly used for code clone detection are: Text-based, Token-based, Tree-based, Program dependency graph-based, Metric-based. In paper we review the various techniques for detecting the code clones.

I. INTRODUCTION

Code cloning means coping the some segments, variables, functions and then paste them in another program is known as code cloning. In every software almost 7% to 23% code is copied. This make the software redundant and its maintenance cost is increases. especially many open source code are commonly copied. So detection of these clones is very necessary to maintain the software cost. The various detection techniques are used on basis of which type of code clone are present in the software.

TYPES OF CODE CLONE

There are 4-types of code clones.

TYPE-1: Identical clones in which only the white spaces, comments and may layout can also vary.

TYPE-2: Structurally/Syntactically identical except for variations in identifiers, literals, types, layout and comments.

TYPE-3: Copied fragments with further modifications. In this the statements can be added or removed in additions to layout, comments and syntax.

TYPE-4: In this type the functionality is same. Two fragments perform the same computation but implemented through different syntax.

The Type-1, Type-2, Type-3 code clones are known as syntax clones whereas Type-4 is called semantic clone.

A. Clone Pair:

Clone pair: if there is an equivalence relation between two code segments, then they form a clone pair.

B. Clone Class

Clone Class: It is defined as collection of similar code segments. Each code segment in a clone class form a clone pair with other code segment of that class.

C. CODE CLONE TERMS

Exact Clones: Two or more code fragments are called exact clones if they are identical to each other with some differences in comments and whitespace or layout.

Renamed Clones: People use the term renamed clones when identifier names, literals values, comments or whitespace changes in the copied fragments. Thus, a renamed clone is essentially a Type II clone.

Parameterized Clones: A parameterized clone or p-match clone is a renamed clone with systematic renaming. The clone detector looks for consistent name matching rather than normalizing all identifiers and/or literals to a especial symbol. Parameterized clones are thus a subset of Type II clones.

Near-Miss Clones: These are those clones where the copied fragments are very similar to the original. Editing activities such as changing in comments, layouts, changing the position of the source code elements through blanks and new lines, changing the identifiers and literals.

Gapped Clones: A gap clone code is partly similar to the original segment. In this type of clones, there is some different code portion between the segments. This different code portion is known as a gap

Structural clones are simple clones within a syntactic boundary following syntactic structure of a particular language. These boundaries can be function boundary, statement boundary, class boundary etc. depending on the programming language of interest.

II. RELATED WORK

Various techniques can be found in the code clone literature to detect code clones. Along with the techniques there are number of tools available for various languages. [1][3][4][8][9][10].

CODE DETECTION TECHNIQUES

A. TEXT- BASED APPROACH

Text-Based Approach: In this approach the two strings are compared from the source. From the comparison the clone pair and clone classes are found. In this no or little transformation is applied. But the comments and white spaces are removed from the code and hence normalization can be applied on the source code. Baker's Dup uses a sequence of lines as a representation of source code and detects line-by-line clones in the Text-based approach [6].

B. TOKEN-BASED APPROACH

Token-Based Approach: In this approach the tokens are generated from the source code. The entire source system can be lexed /parsed /transformed to a sequence of tokens.

From the generated tokens the duplicated code can be detected. CCFinder of Kamiya et al. have developed a code clone detector for various languages using the token-based approach [6].

C. TREE-BASED APPROACH

Tree-Based Approach: In this a program is parsed to a parse-tree or abstract syntax-tree(AST) with a parser of the language of intrest. Similar sub-tree are searched by using any of the tree based technique and duplicated code are detected from these sub-trees. The parse or AST tree contains the complete information about the source code. Koschke et al discuss a technique that uses a suffix tree to identify clones. In their technique first AST is generated which is serialized and then suffix tree are formulated. The technique helps in detecting the Type-1 as well as type-2 clones[5].

D. PROGRAM DEPENDENCY GRAPH-BASED APPROACH

Program Dependency Graphs-Based Approach (PDGs): PDGs represent the structure and data flow within the program. In this approach we try to identify the similar sub-graphs from the source code. Identified sub- graphs can be mapped back onto to the program and presented to the user. Komondoor and Horwitz PDG-DUP which finds the isomorphic PDG sub-graphs using the program slicing. Whereas Krink uses an iterative approach for detecting maximal similar sub-graphs[6].

E. METRIC-BASED APPROACH

Metric-Based approach: This approach calculates the metric from the source code and uses these metrics to measure the clone in the software. Rather than working on source code directly this approach uses metrics to detect the clones. It gathers the different metrics for code fragments and compare these metrics vectors instead of comparing code directly. Mayrand et al. calculate several metrics for each function unit of a program. Units with the similar metrics values are identified as code clones[6].

III. PARAMETERS USED FOR CODE CLONE DETECTION TECHNIQUES.

There are several code clone detection techniques.

The comparison of these techniques is much worth to pick the right techniques for the problem statement. Different parameters have been chosen for comparison of five techniques (text-based, token-based, tree-based, graph-based, metric-based).These parameters are also known as the clone challenges.

Some of the parameters are stated below:-

- *Portable:*-The techniques should be portable for multiple languages and dialects. As many programming languages are used with several dialects. It is expected that clone detection techniques should be easily portable and configurable for different languages.
- *Precision:*-The techniques should be sound enough to detect the less number of false positives. It is also being said that techniques should find the duplicated code with higher precision.
- *Recall:*-The techniques should be capable of finding the clones that are used for system interest i.e. according to the software project.
- *Scalability:*-It is difficult to find the clones of the code from the large and complex system. The techniques would be scalable to handle the large and complex system with efficient use of memory.
- *Robustness:*-A good technique should be robust for different editing activities that are applied on copied fragment and detect the clones with higher precision and recall.

ANALYSIS OF VARIOUS DETECTION TECHNIQUES

As we have seen that there are various code clone detection techniques which are used to detect the clones of the code from the original one. There are various parameters stated to choose the right technique and tools. We see that text-based techniques (line-based and parameterized line-based in the table) are easily adaptable to different languages. Token based techniques use suffix tree algorithm to detect the clone and also break the code into the token. The Tree-based techniques look at the structural properties of the source code. The PDG-based techniques use the complex graph and flow among the codes which are complex. In the Metric-based technique first the metric are detected than other parameters are calculated from these metrics.

S. no.	Name of techniques	Computation-al complexity	Precision-n	Recall-l
1	Text-based	Depends on algorithm	High	Low
2	Token-based	linear	Low	High
3	Tree-based	Quadratic	High	Low
4	PDG-based	Quadratic	High	Medium-m
5	Metric-based	linear	Medium	Medium-m

Fig.1

IV. CONCLUSION

Many techniques are used to detect the code clones. This paper provides a brief overview of various techniques. No single technique is good for all type of code clone detection. Token-based approach has medium portability, low Precision, high Recall and high scalability whereas the Metric-based approach has low portability, medium precision, medium Recall and High Scalability. In the future we will combine both the approaches for higher accuracy to detect the clone of the code and simulate it on the VB.NET by using the source code of C\C++. Therefore this area provides a lot of opportunities for research.

REFERENCES

[1] Lifang Han1, Baojiang Zhang1,Zhongxian Li, Jianxin Wang2, Yongle Hao 3CNITSECBeijing, China.
[2] Dep. of Electronic and Information Engineering Kumamoto National College of Technology 2627 Hirayama-Shinmachi, Yatsushiro, Kumamoto, Japan.
[3] Yang Yuan and Yao Guo Key Lab of High-Confidence Software Technologies (Ministry of Education) Department of Computer Science, School of EECS, Peking University Beijing 100871, P. R. China.
[4] Geetika Bansal, Rajkumar Tekchandani Computer Science and Engineering Thapar University Patiala, India.
[5] CODE CLONE DETECTION A NEW APPROA-CH- Sanjeev Chakraborty
[6] Chanchal Kumar Roy and James R. Cordy September 26, 2007.
[7] Filip Van Rysselberghe, Serge Demeyer University Of Antwerp Middelheimlaan 1, B 2020 Antwerpen
[8] Hiroaki Murakami, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki and Shinji Kusumoto Graduate School of Information Science and Technology, Osaka University, 1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan.
[9] Kanika Raheja, Rajkumar Tekchandani CSED, Thapar University, India.
[10] Florian Deissenboeck, Benjamin Hummel, Elmar Juergens Institut für Informatik, Technische Universität München

