A Novel Approach to Detect Code Clone

[1]Alfiya Pathan, [2]Pragati Rajure, [3]Pooja Devkar, [4]Ankita Charahate

Student at PDEA's College of Engineering Pune

***Abstract*: In software development process, coping of existing code fragment and pasting them with or without modification is a frequent process. 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. Detection of clones in a large software system is very tedious tasks but it is necessary to improve the design, structure and quality of the software products. Clone detection techniques first analyses the source code, represent the code in their proposed ways such as text, tokens, Abstract syntax tree etc. and then perform matching algorithms to detect the clones. Various techniques and tools have been proposed for detecting clones. Each of these techniques have their own merits, but are not useful in all the scenarios where a code can be cloned.**

***Index Terms*: Code clone, Software code, Clone detection, Semantics clone, Detection technique.**

I. Introduction

When the source code is copied and pasted or modified, there will be a lot of identical or similar code snippets in the software system, which are called code clones. Because code clones are believed to result in undesirable maintainability of software, numerous approaches and techniques have been proposed for code clone detection. However, most of them are based on the source code, while only a few employ the bytecode to detect code clones. The code quality analysis (improved quality code), replication identification, virus recognition, facet mining, and bug exposure are the other software engineering tasks which require the mining of semantically or syntactically identical code segment to facilitate clone detection significant for software analysis [1]. Fortunately, there are a number of comparison and evaluation studies which are related to numerous clone detection techniques. Recently, Rattan et al. [2], has presented a methodical survey on clone detection while Roy et al. [3] has presented an qualitative comparison and evaluation of clone detection tools and techniques. Bellon et al. [4] has presented an extensive quantitative assessment of six clone detectors which is based on large C and Java programs for clone detection. Further, the potential studies have evaluated the clone detection approach in other context[5]. There are many software clone detection techniques and tools that differ from each other on the basis of approach used by them to detect clones. Cloning between two program codes is recognized on the basis of textual similarity and functional similarity. These types of similarities define the clone type[6]. Based on the textual similarity we define the type 1, type 2 and type 3 clones.
Type 1 (Exact clone):- Identical code fragments except for variations in white spaces, layout and comments.
Type 2 (Renamed/Parameterized Clone): Syntactically same code fragments except for changes in identifiers, literals, types, whitespaces, layouts and comments.
Type 3 (Near miss clone):- Copied with further modification such as "change, add or remove" statements in addition to changes in identifier, literal, types, whitespaces, layouts and comment.
Based on the functional similarity we define type 4 clones imitated as a semantic clone.
Type 4 (Semantic clone):- Code fragment which are functionally similar but not textually similar.

II. Methodology

A clone detector must try to find pieces of code of high similarity in a system's source text. The main problem is that it is not known which code fragments may be repeated [7]. Thus the detector really should compare every possible fragment with every other possible fragments. Such a comparison is prohibitively expensive from a computational point of view and thus, several measures are used to reduce the domain of comparison before performing the actual comparisons [8]. Even after identifying potentially cloned fragments, further analysis and tool support may be required to identify the actual clones.
A) Preprocessing: At the beginning of any clone detection approach, the source code is partitioned and the domain of the comparison is determined. There are three main objectives in this phase:
o **Remove uninteresting parts:** Whole source code uninteresting to the comparison phase. Those are filtered out in this phase. For example, partitioning is applied to embedded code to separate different languages (for example, SQL embedded in Java code, or Assembler in C code).
o **Determine source units**: After removing the uninteresting code, the remaining source code is partitioned into a set of disjoint fragments called source units. These units are the largest source fragments that may be involved in direct clone relations with each other. Source units can be at any level of granularity, for example, files, classes, functions/methods, begin-end blocks, statements, or sequences of source lines.
o **Determine comparison units**: Source units may need to be further partitioned into smaller units depending on thecomparison technique used by the tool. For example, source units may be subdivided into lines or even tokens for comparison. Comparison units can also be derived from the syntactic structure of the source unit.
B) Transformation: Once the units of comparison are determined, if the comparison technique is other than textual, the source code of the comparison units is transformed to an appropriate intermediate representation for comparison.
C) Normalization: Normalization is an optional step intended to eliminate superficial differences such as differences in whitespace, commenting, formatting or identifier names.

D) Match Detection: The transformed code is then fed into a comparison algorithm where transformed comparison units are compared to each other to find matches. The output of match detection is a list of matches in the transformed code which is represented or aggregated to form a set of candidate clone pairs. Each clone pair is normally represented as the source coordinates of each of the matched fragments in the transformed code.
E) Formatting: In this phase, the clone pair list for the transformed code obtained by the comparison algorithm is converted to a corresponding clone pair list for the original code base. Source coordinates of each clone pair obtained in the comparison phase are mapped to their positions in the original source files.
F) Post-processing: In this phase, clones are ranked or filtered using manual analysis or automated heuristics. Aggregation: While some tools directly identify clone classes, most return only clone pairs as the result. In order to reduce the amount of data, perform subsequent analyses or gather overview statistics, clones may be aggregated into clone classes.

III. LITERATURE SURVEY

Sr.no	Paper Name	Author Name	Conclusion
1	A Approach for Detecting Type-IV Clones in Test	CoNovelde Brent van Bladel, Flanders Make vzw Belgium	It detect type 4 as well as type 3,2,1
2	CloneTM: A Code Clone Detection Tool Based on Latent Dirichlet Allocation	Sandeep Reddivari Mohammed Salman Khan	Support a variety of programming languages and adopt different clone detection strategies at different levels of complexity
3	Detecting Java Code Clones Based on Bytecode Sequence Alignment	Dongjin Yu , (Member, Ieee), Jiazha Yang, Xin Chen, And Jie Chen	It detects data clone of type 3,2,1.
4	A Systematic Review on Code Clone Detection	Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam.	It gives information on all different types of data clone types,and its tools.

IV. CONCLUSION

The main reason behind code cloning is copy and paste activity done by programmer. As code clones lead to Increase in maintenance cost and bug propagation, there is a need to remove them. On the other hand, practice of cloning in programming can increase program productivity and helps in plagiarism detection. So, there is a need to study the clones properly and categorize them in to harmful and useful clones. Harmful clones can be removed and useful ones can be retained by refactoring them using appropriate technique. There are different types of clones that exist in the source code and number of techniques to detect them has been proposed by different authors from time to time. In this paper, we have focus on code cloning process and tried to give a brief review of key areas related to clones that will help researchers to get started with clones. Comparative review of various clone detection techniques summarized in the form of a table will help the researchers in selection of appropriate technique according to their needs. Type 1 and 2 clones are easy to identify as compared to type 3 and 4 clones. So, there is a need for techniques which can detect type 3 and 4 clones efficiently.

V. ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my guide DR. R.V. Patil , principal of P.D.E.A. college of engineering ,manjari as well as my teacher prof. Shinde sir who gave us golden opportunity to do this wonderful project on the topic of data clone in codes .

References

[1] R.V.Patil "Software Code Cloning Detection and Future Scope Development -Latest Short Review " ICRAIE-2014
[2] XIN CHEN "Various Code Clone Detection Techniques and Tools ". Program Compression (ICPC)2018 IEEE 19th International Conference on IEEE .
[3] Sandeep Reddivari " Clone Types Based Comparison " Software Maintenance and Re-engineering (CSMR)IEEE 2018.

[4] Kamiya Toshihiro "Classifying Code Clone With Configuration" proceeding of the 4th International Workshop on Software Clones ACM 2010.
[5] Roy CK, Cordy JR, Koschke R. “Comparison and evaluation of clone detection techniques and tools: A qualitative approach. Science of Computer Programming “2009; 74:470–495.
[6] Roy CK, Cordy JR. A survey on software clone detection research. TR 2007-541, Queen’s School of Computing, 2007; 115
[7] D.Gayathri Devi, Dr. M. Punithavalli “Comparison and Evaluation on Metrics based Approach for Detecting Code Clone” in IJCSE, ISSN: 0976-5166 Vol. 2 No. 5 Oct-Nov 2011 page no- 750.
[8] Jean Mayrand, Claude Leblanc, Ettore M. Merlo Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics” ICMS 1996 1063-6773.