IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume4, Issue-8 E-ISSN: 2347-2693

Various methods for Measuring Similarity and code clone detection

Gagandeep Kaur[1*] and Bikrampal Kaur

[1,2] Chandigarh Engineering College, India

Available online at: www.ijcseonline.org

Received: 24/Jun/2016 Revised: 10/Jul/2016 Accepted: 16/Aug/2016 Published: 31/Aug/2016

Abstract - Code clones means duplicate fragments of source code, have been identified as "a major source of faults, which means that duplicating can be a considerable problem during development and maintenance". As a consequence, a large body of planned has been industrialized on how to prevent, or spot and remove code clones. The problem with code clones is that they are related only by their resemblance, i.e., implicitly rather than explicitly which makes it difficult to notice them. Therefore, changes like promotions or patches that are often meant to affect all clones in a similar way are frequently not applied to all of them uniformly. Code clone helps the developers from probable mistakes, to save time and exertion in planning the logic, to help in decoupling of classes or components and more important it reduces development cost. But identical code is generally considered as unwanted for number of reasons. Introduction of bad design and lack of good legacy structure or concept may be caused due to code clones. Probably the biggest problem in model clone detection is defining exactly what a model clone is just as for code clones a small part of a domain model of the "Library Management System".

Keywords: Code Clone, source code, duplicate fragments, problems and domain model.

I. INTRODUCTION

A code clone is a code helping in basis files that is identical or similar to another [1]. Duplication of code occurs often during the growth of large software systems. Code cloning is a form of software reuse, and exists in nearly every software scheme.[2] This ad-hoc form of reuse consists in copying, and eventually modifying, a block of current code that tool a piece of required functionality. Duplicated blocks are called clones and the act of repetition, including slight alterations, is said cloning. The results of several studies indicate that a substantial fraction (5-10%) [3]of the basis code in large software systems is duplicate code. Software clone is usually made by programmer's copy and paste doings. Programmers often copy and paste an existing similar code and further modify it according to their need. Code cloning or the act of copying code wreckages and making minor, non — functional changes, is a well-known problem for emerging software systems leading to repeated code fragments or code clones. The normal operative of the system is not pretentious but further development may become prohibitively expensive [4].

In this section, we survey the code clone detection means code clones are fragments of code that are very similar text, syntax and String. There is common phenomenon in a [5] request that has been under development for some time. Clone makes it hard to change your request since you have to find and change more than one fragment. Why code duplication and various types of techniques to identify the code clone detection.[6]

II. WHY CODE DUPLICATION?

There are a number of reasons why designers clone source code. Cloning mostly occurs because computer operator fined that it is cheaper and faster to use the copy and paste feature than writing the code from scratch. Sometimes programmers intent on implementing new functionality find some employed code that does a calculation nearly identical to the one wanted copy it entirely and then adapt in place [7]. While this is really good reuse repetition, it complicates the upkeep process. Code cloning is careful a serious problem in manufacturing software. [8]Repeated code proves easy and inexpensive during the software expansion phase, but it makes software upkeep much firmer. Software clone has a number of undesirable effects on the excellence of the software. Besides snowballing the amount of the code, which needs to be upheld, it also upsurges the bug probability[9]. So there is a need to detect the clones to figure out the malfunctions and to help healthier software understandability and maintenance. Concerning the detection of duplicated code, numerous techniques have been positively applied on industrial schemes. These techniques can be roughly classified into following categories [10]:

- *String-based,* i.e. the program is alienated into a number of strings (typically lines) and these strings are compared in contradiction of each other to find orders of duplicated strings.

- *Token-based,* i.e. a lacer tool divides the program into a watercourse of tokens and then searches for series of alike tokens.
- *Syntactic-based* approaches use a parser to convert source[6] program into parse trees or abstract syntax tree matching or metrics to find clones.
- *Parse-tree based,* i.e. after building a whole parse-tree one achieves pattern matching on the tree to search for similar sub—trees.
- *Metric-based,* i.e. metrics are calculated from program and these are used to find duplicated code.
- *Hybrid-based,* i.e. detection techniques that use a combination of other clone detection techniques.
- *PDG based,* i.e. after obtaining program dependency graph similar graphs are search.

III. TYPES OF CLONE

Code clone could be of any sort that all rely on upon the developer's technique and aptitude of utilizing the code which differs from replicating as it is to duplicate the code though with some change which would be complete at diverse level in the technique. In software system code pieces predominantly demonstrates two sorts of similarities. They are said to be comparable if their code content matches or they can be relative on their functionalities bases if the conduct among them coordinate. Primarily clones are of four kinds out of which first three sorts are under textual similarity and the last sort is under functional similarity. One clone type of similarity considers textual similarity), and other second considers the semantic level that the clone code essential to have the identical behaviors, means the functional similarity.

A. Textual Similarity: Two code wreckages can be similar based on the resemblance of their program text we differentiate the subsequent sorts of clones. The subsequent types of clones are deliberated in order to find textual similarity [7].

Type I

In Type I clone, a copied code fragment is the same as the original. However, there might be some variations in whitespace (blanks, new line(s), tabs etc.), remarks and/or designs. Type I is widely known as exact clones

Type II

A Type II clone is a code piece that is the same as the unique except for some possible variations about the corresponding names of user-defined identifiers (name of variables. coefficients, class. methods and so on) layout, identifiers, remarks, literals, and sorts. The specific reserved words and the sentence structures are essentially the same as the original one.

Type III

Type DI is copy with further modifications. E.g. a new statement can be added, or some statements can be detached along with various dissimilarities in layout, identifiers, remarks, literals, and sorts. The structure of code piece may be changed and they may smooth look or behave slight differently. This kind of clone is hard to be discovered, for the reason that the wholly framework understanding is needed.

Type IV

Type IV clones are the results of semantic similarity between two or additional code fragments which could accomplish the same computation however actualized through diverse syntactic variations. In this category of specific clones, the cloned part is not necessarily copied from the first one. Two code fragments may possibly be established by two different programmers too.

IV. RELATED WORKS

HaraldStörrle et.al, 2015 [1] this paper described as, Code Duplicates are a main source of software faults. Thus, it is probable that model duplicates have a significant adverse impact on model excellence, and thus, on any software shaped based on those models, notwithstanding of whether the software is made fully automatically or hand crafted following the drawing defined by the model. Inappropriately, however, model clones are much less well deliberate than code clones. In this paper, presented a clone detection process for UML domain models. A method covers a much better variety of model types than present approaches while providing high clone detection rates at high speed. Jian Chen et.al ,2015 [2] In this paper, examine the use of a clone sensor to classify known Android malware. They assemble a set of Android submissions known to comprise malware and a set of kind applications. They extract the Java source code from the double code of the submissions and use NiCad, a near miss clone detector, to invention the classes of clones in a small separation of the malicious presentations. Then use these clone programs as a signature to find related source files in the rest of the hateful applications. The benign gathering is used as a control group. Mr. Ritesh V. Patil et.al,2014[3]examined existing code in software development life cycle. Although code cloning is a suitable way for designers to reuse current code it could possibly lead to negative influences, such as code size needlessly increased and may lead to unused, dead code. There are numerous clone detection techniques based on dissimilar comparison parameters. Discovered clone detection tools and methods do not sufficiently satisfy with regards to rapidity and correctness. Ritu Garg et.al,2014[4] This paper offers a brief impression to the detection of these risk and contradictions in either of the two stages of software development system i.e. Design phase or the operation phase along with their experts and frauds. Ritesh V. Patil et.al,2014 [5] described as, the clone discovery consequences for a single source code variety gives a developer with particulars about a discrete state in the development of the software system. However, tracing clones through numerous source code versions enables a clone investigation to take into replication a temporal dimension. This nice of an investigation of clone evolution may be utilized to find out the outlines as well as features displayed by clones as they evolve within a system. Developers may apply the consequences of this analysis to recognize the clones more methodically, which may guide them to handle the clones more automatically. Later, studies of clone development provide significant role in observing as well as handling disquiets of cloning in software.

IV. TECHNIQUES OF CODE CLONE DETECTION

Token based clone detection approach takes source code and converts them in lexemes/tokens. From order of tokens, token watercourse is formed. The heart of token based matching approach is how to use syntax tree and grammar array. Some famous out of these tackles are dup[6,7] that uses token sequence and used them as syntax tree, CCFinder[6,8] uses suffix tree matching technique

Tree-based techniques find clones by finding similar sub trees. Variable names, verbatim values and other leaves (tokens) in the foundation may be abstracted in the tree representation, allowing for more sophisticated detection of clones. A compiler generator is used to generate a constructor for annotated parse trees. Sub leaves are then hashed into loads. Only within the same bucket, sub trees are compared to each other by a tolerant tree corresponding. The hashing is elective but reduces the amount of necessary tree comparisons drastically.

Metrics-based methods gather a number of metrics for code wreckages and then compare metrics vectors rather than code or ASTs directly. One general technique includes fingerprinting functions, metrics calculated for syntactic units such as a class, function, way and statement that harvest values that can be compared to find clones of these units. In most cases, the basis code is first analyzed to an AST or control flow graph (CFG) on which the metrics are then calculated. Use numerous metrics to identify purposes with similar metrics [9] values as code clones. Metrics are calculated from names, layout, languages, and (simple) control flow of purposes. A function clone is identified as a pair of whole function bodies with similar metrics values

V. IMPORTANCE OF CODE CLONE

Code clones decreases program comprehensibility, and maintainability. Overall, it is beneficial to discover code clones to improve quality of the software systems. In huge software system, generally 10-15% of source codes are cloned [10]. To save programming efforts as well as time, the copy pasting is used. Code clone detection is essential in order to utilize storage resources, maintain software and improve code productivity. Programming size increases for no reason. Code replication increases the overhead software maintenance, since bug introduction in the source may be replicated accidently or unknowingly.

VI. CONCLUSION

Cloning of code has become one of the easiest ways to complete a project, who does not want to invest their time on doing programming their project. It's a loss for those who really work hard for the project coding. The date no such method has present who can evaluate the cloning for several languages with one piece of code. The purpose research work has overcome the drawbacks of the previous attempts by removing the bar of the language which follows the architecture of C++.

REFERENCES

[1] Störrle, Harald. "Effective and Efficient Model Clone Detection." Software, Services, and Systems. Springer International Publishing, 2015. 440-457.

[2] Chen, Jian, et al. "Detecting Android Malware Using Clone Detection."Journal of Computer Science and Technology 30.5 (2015): 942-956.

[3] Wyss-Coray, Anton, et al. "Biomarkers of aging for detection and treatment of disorders." U.S. Patent Application No. 13/575,437.

[4] Ritu garg, et al. "Code Clone v/s Model Clones: Pros and Cons." International Journal of Computer Applications (0975 – 8887) Volume 89 – No 15, March 2014.

[5] Patil, Ritesh V., et al. "Software code cloning detection and future scope development-Latest short review." Recent Advances and Innovations in Engineering (ICRAIE), 2014. IEEE, 2014.

[6] B. Baker. "Finding Clones with Dup: Analysis of an Experiment." IEEE Transactions on Software Engineering. vol. 33. no. 9. pp. 608-621. 2007.

[7] B. Baker. "On Finding Duplication and Near-Duplication in Large Software Systems", in Proceedings of the Second U'orking Conference on

Reverse Engineering (WCRE 195). pp. 86-95. Toronto. Ontario. Canada. July 1995.

[8] C .K. Roy. J.R. Cordy and R. Koschke, “Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach." Science of Computer Programming, vol.74. no. 7. pp. 470-495. May 2009.

[9] Chao Liu. Chen ChenJiawei Han and Philip S. Yu.,"GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis", In the Proceedings of the 13”’ ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 872-881. Philadelphia. USA. August 2006.

[10] EttoreMerlol. "Detection of Plagiarism in University Projects Using Metrics- based Spectral Similarity." In the Dagsmhl Seminar: Duplication, Redundancy, and Similaritv in Software. 2007.