A Novel Technique of Software Clone Detection based on Evaluation Parameters

Roopam[1], Pratyush Shukla[2], Mohit Arora[3*], Shivali Chopra[4]

School of Computer Science and Engineering,

Lovely Professional University. Phagwara, Punjab, India.

Abstract

Code clones are novel subtle way to deal with issues pertaining from one territory module to another fragment or module. They are increasing the rate of peril in bug duplication within each code that has a copy, whenever there was a bug in wellspring of clone. The clone in code is equivalent to or indistinct code in the original source code that is made either by reproduction or a couple of changes. Clone is an indefatigable sort of programming reuses that effect on help of enormous programming. Till now, the researchers have underlined on perceiving type 1, type 2, and type 3 sort of clones. The present code clone area devices are utilized to perceive source code cloning present within them. Throughout this paper, a review is done on programming clone area to assess cross breed strategy subject to various parameters. Rather than using single technique on the code, creamer methodology is being used, suggests two strategies are being solidified together. We can achieve increasingly essential adequacy, higher accuracy and execution for code clone recognizable proof as opposed to using single strategy.

Keywords: Code clone, parsing, unoriginality, refactoring, re-use, semantic, syntactic closeness.

1. Introduction

During the case of programming structuring principal, one need to ensure a software must be free from gadget clutters [20]. It recognizes and then by testing or reviewing makes the bugs go away. While developing any artifact to spare time and effort, code of the program is reordered by programming engineer iteratively. In case bugs are present in some module, it expands in each copy. There are a couple of code copies to display and the records for theses copies are inaccessible. This creates a difficulty in turning away these bugs and backing of present programming. Whether the resulting clone begins by including any more value, that is similar yet not indistinct from existing method of reasoning, it is an assumption. Code cloning infers one method where little follow-up of some bits of code is present and a while later it is fixed with or without little minor changes into different bits of the code such that the code can be reused. The code clone is then a stuck segment of the code.

Code cloning faces up as one of the keen elements during the improvement time of programming [5]. As showed by momentum, ask about assessment 7% to 23% of the code within the item system is containing cloned code [6]. It is precarious to the upkeep time onto the deliverable in such a case, that the cloned code involves a bug, to effectively perceive and address a comparative bug through the clones of respective code. It increases the improvement cost in programming and leads to degraded quality programming structure when observing development in programming size.

2. Clone terminologies

- Code section

Code region (several parts of a code) is some movement of lines of codes containing or negating any critics. It is perceived through code area filename, start line of part of code, end line in code piece.

- Code clone

Precisely whenever some code belonging to chronicle two, marks itself different code territory clone of one record.

- Clone pair

If the arrangement of one of the sections of code are undefined from other whether in a proportional report or distinct record, a clone pair is said to be present.

- Clone class

During the course when various parts are indistinct among each other or whether there exists some clone-association amongst them, a clone class is said to be developed.

3. Literature Review

Programming or cloning code expanded into a critical locale of research in current times. Various researchers productively examining such subjects along these lines, using various strategies, have been made to blueprint codes. These follow-ups are grammar based[1], content based [2], diagram based [3] and criterion based [4].

Sonam Gupta et.al in 2014 [5] proposed the existence of several code clone disclosure strategies viz. substance based, token based, hypothetical language structure tree, PDG and metric based in their paper. Various estimations are made dependent on recognizable proof strategy to recognize the 1, 2, 3, 4 types of clones, though some are clones, yet simultaneously none can discover such clone with exactness and viability. A proposed "clone piece count" was made available in this paper, which detects a wide scope of clones inculcating precision and profitability.

Ritesh V. Patil et.al in 2015 [3] proposed in their paper techniques which recognizes duplicate code decline basic undertakings and are implemented to approach the type such as type-3 and several clones imparting testing viewpoints within the assessment. Their goal is to downsize relationship and upgrade exactness, and decentralize the system through the proposed technique. The code decline strategies extensively applied for snappier area of clones.

Geetika bansal et.al in 2014 [4] in their paper proposed various order of estimations for applicability in criterion-based clone acknowledgment. Social occasions of estimations are judged depending on the exactness and audit trials of them. "Check Path" in the proposed approach is used since the precision by it is extended. It tackles type-1, type-2 botches and within gigantic game plan of systems a couple of networks are picked that contain lesser relationship.

Prajila Prem et.al in 2013 [9] in their paper suggests a methodology wherein some bit of code has been studied and fixing it later with or without minor changes into other code sections. The stuck code divide is referred as a code clone. Throughout respective paper distinctive code clone recognizable proof methodologies, contraptions for acknowledgment approaches and the system applied for them and the assessment in code has been discussed. The artistic procedure displays an unforgiving outline of the code duplicates which are convenient to receive adamantly the token-based system gives a scrupulous vision of supplied parts of replicated code and amazingly opposes rename undertakings. Syntactic systems skilled with duplicate subroutines revelation, are autonomous of little opposites, hence it performs on top with blending with refactoring instruments that apply this technique level.

Mena bharti et.al in 2014 [6] in their paper proposed fit code clone area frameworks, merits, shortcomings, utilization of code clones on account of the closeness of code clone.

Kuldeep Kaur et.al in 2015 [2] in their paper proposed that for saving time and labor, program code is reordered iteratively. A bug in some module is carried and reproduced in subsequent copies. Code cloning suggests a system wherein we following some code sections and progressively fix it having some or none of the minor changes into other sections of the code, such that the particular code section can be reused. This so-called code clone is then a stuck code piece. It has been seen that substance-based framework can perceive simply Type 1 clone. Token based method recognizes Type1, Type II clone. Tree based system distinguish Type1, Type II, Type III clone, program dependence diagram approach is used for perceiving Type IV clones yet it is difficult to develop a sentence structure tree since its capriciousness is especially high. It has been seen that dependent on exactness, audit, life and flexibility no strategy is found perfect.

Robin Sharma et.al in 2013 [8] in their paper observes within the source code a judicious clone utilizing object orchestrated point of view. This paper portrays the novel approach which utilizes two frameworks: - unique method and metric based system to perceive code clone present within open source structure. The procedure is showed up diversely in connection to the present device with view clone in the thing and provide exact outcome, something less incredible.

Rubal sivakumar et.al in 2012 [10] in their paper recognizes a wide range of clones in web application by utilizing various hybrid approaches. This strategy presents an effective method for clone location. Author aim to discover utilitarian clone in web application and then kill the copy code within it thus improving the inadequately structured web applications.

Bayu priyambadha et.al in 2014 [7] in their paper discussed primary worries that how to discover, present in void, the information, yield and impacts and perform strategies negating the parameters. It is applied utilizing PDG. The utilizing of semantic clone discovery strategy shall build accusation of recognition around 89%.

4. Types of Clones

a. Type 1

It refers to clone with indistinct copy. Such clones of the code are unclear in identity. Type 1 clones oversees clear territories and comments. [6]

b. Type 2

Such clones of the codes are equitably indistinct. Modification of literals takes place[8]. For instance, elements identities and limits. Type 1 and Type 2, hence, become indistinguishable.

c. Type 3

Such clones in codes are imitated partly by removing or including and traded lines [9]. Type 3 code clones are recognized by PDG.

d. Type 4

Such clones of the codes are unmistakable in sentence structure anyway helpfulness and genuine is same. The recognition of these Type 3 clones is arduous. These occur unintentionally. [3].

5. Merits of code cloning

The main highlights for programming cloning are [6]:

a. Research Growth: Programhming ID strategies are adequately used in programming advancement assessment by looking powerful thought of different clones in different variations of a structure.
b. Aides in Code Compression: Location of code clone techniques can be used by decreasing the source code size.
c. Aides in Program learning: In the unlikely event, that the value of cloned part is comprehended, it is possible to have a general idea on various records containing copies of that segment of code.

6. Demerits of code cloning

The different drawbacks for programming cloning are:

a. Increase in Bug Propagation: In an unlikely event, a bug is present within a code segment which is reused by adjusting and staying containing or negating minor alterations,
b. Boost the Probability occurring in Bad Design: In further undertakings it will get hard to reuse that piece of the implementation.
c. Boost the Maintenance Cost: There will be necessity to discover comparable bug in all cloned parts which is a troublesome assignment.
d. Amplification in Probability of discharging a New Bug: The skeleton of cloned code is reused in which programming designer adjust the code according to slanting needs.

7. Application of code cloning

The different applications for programming cloning are:

a. Recognition of Plagiarism and Copyright Fragments: Detecting written falsification and copyright in sections and finding comparable code.

b. Discovers Usage Patterns: The practical utilization structure of a pattern can be found if all the recreated sections of an equivalent source part can be identified.

c. Distinguishes vindictive programming: It is conceivable to discover the proof where parts of one programming framework coordinates with framework of another by contrasting one noxious programming family with another.

8. Techniques of code clone

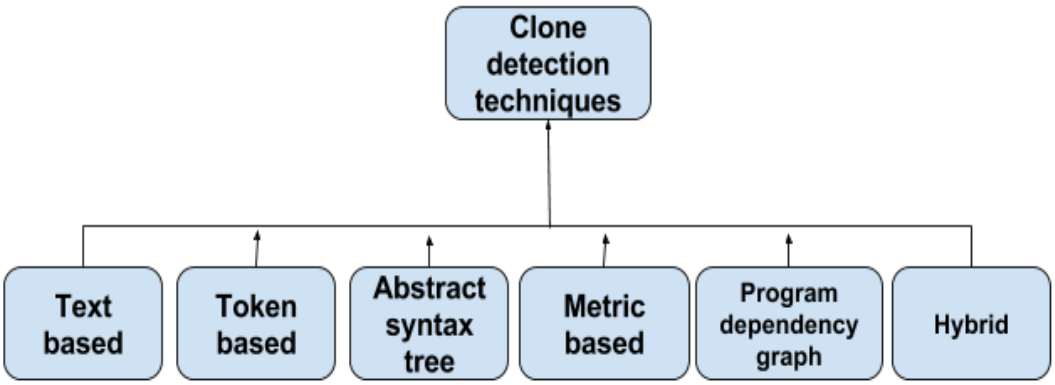

Figure 1. Techniques for unearthing clones

A. Text based technique for detecting clones

It needs least change. In this, fragment of code is considered as groupings of strings and a short time later these are co-related with each other in order to find a comparative string. It is moreover called as string-based procedure. Iteratively line relationship shall be performed on both code pieces [10]. Whether printed similarity is found amongst them, then the perception about them is as clones and is depicted in Figure 2.

B. Abstract-syntax tree-based clone detection technique

In this, code clones are looked by means of checking for same or similar sub trees which implies proximity of code clone [6]. The level of precision is perfect yet it achieves flimsy adaptability as it depends upon the estimation that is being used to make (manufacture) and relate of the trees.

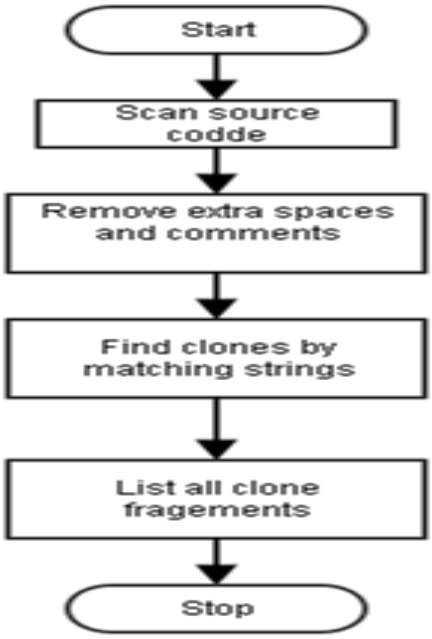

Figure 2. Text Based Technique

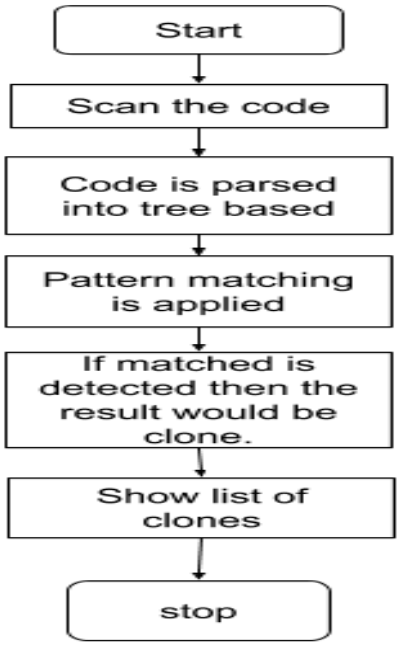

Figure 3. Abstract Syntax Tree Technique

C. Token Based Clone Detection Technique

It is generally called lexical outlook. This philosophy uses parser for the difference in source code into a plan of tokens. Parameterized organizing with postfix trees constitutes one of the strategies for token-based approach [5].

D. Graph Based Technique

This technique highlights dependence on the data and control stream. After executing, PDG estimation is exploited to recoup clones, which aid in findings from the clones. The dependence graphs ought to have been development in this procedure. PDG based disclosure outlook is rather incredible as it can recognize non-neighboring code clones [7]. In any case, it is costly system.

E. Criterion Based Technique

Criterion is approached to evaluate clones in programming post the tallies of estimations after the source code [4]. This procedure parses source code in accordance to the depictions of AST/PDG so as to tally the criterion. This approach gives high precision and adaptability level.

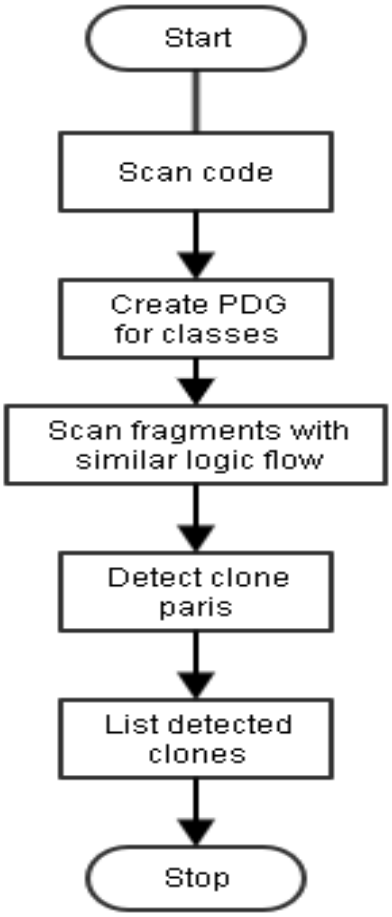

Figure 4. Graph Based Technique

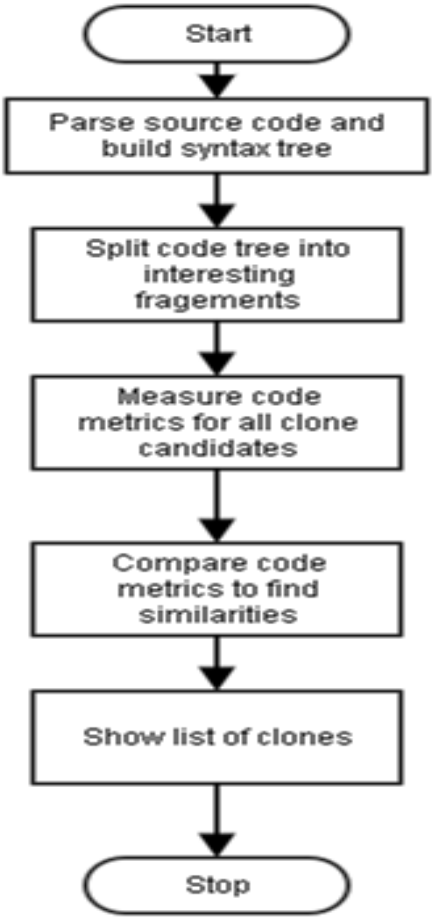

Figure 5. Metric Based Tree Technique

F. Hybrid clone Technique

By mixing and using at any rate two recently referenced strategies clones can be perceived. This framework holds ideal impetus over normal system. For example: graph and estimations framework are employed hybridized for best results.

9. Clone Detection Process

Some regular advances drew in with recognizing clones whether they are real clones or not. This strategy is exorbitant, requires brisk count speed. In light of comparability, clones are observed and detected amongst the clone sets. This is basic code clone process.

The same has been depicted in Figure 6.

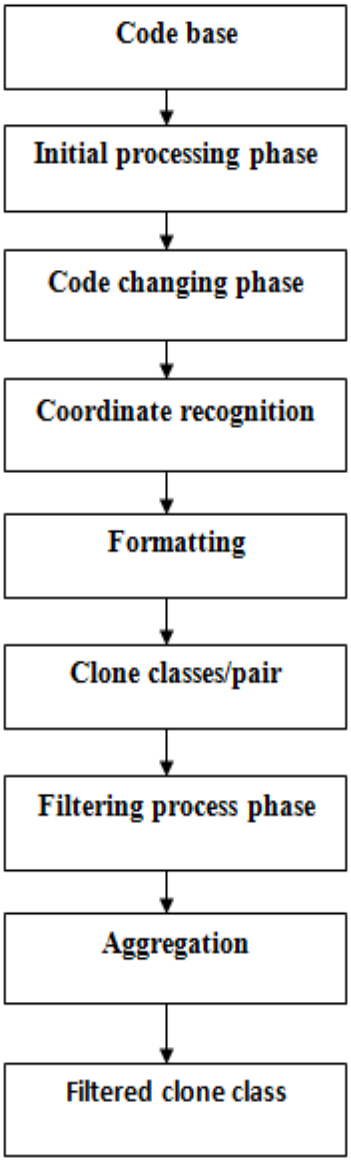

Figure 6. Clone Detection Process

A. **Handling stage:** This stage seeks after two phases one is parceling the source code into the territories generally called division. Furthermore, comprehend the locale of relationship. There are certain objectives of this stage [6]:

1. End of undesirable parts.

2. Make sense of source units.

3. Make sense of examination units

B. Transformation phase: This phase has the main goal to transit the source code units into peculiar intermediate representations and the operation is called as extraction. This step is further subdivided into following

1. Extraction.

2. Tokenization.

3. Parsing.

C. Coordinate recognition: Transformed code obtained through modifications from the previous steps deployed onto the comparison algorithm wherein all of the transformed comparison units are gauged on the basis of distinctness to segregate the matches. Thus, we are provided with a set of candidate clone pairs.

D. Formatting: The clone pair list for the altered code acquired by the comparison algorithm is modified into a relating clone pair list for the original code base.

E. Filtering process phase: It further contains two parts:

F. Manual analysis: The false positives, through human experts, are weeded out.

G. Automated heuristic: Through the filtering process, few parameters are set accordingly already.

H. Aggregation: For the purpose of expelling information with a specified goal tied to it, a subsequent examination is performed such that clones can be wrapped into clone classes.

11. Proposed flowchart

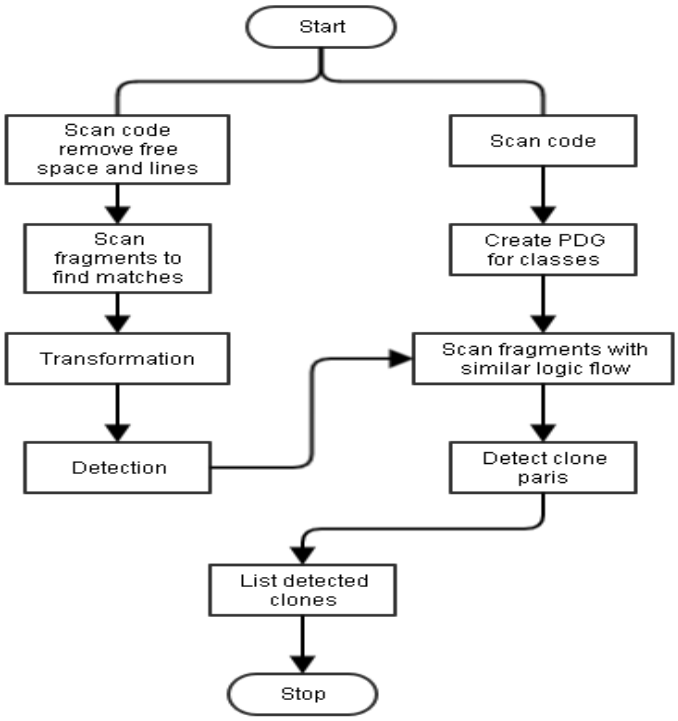

Figure 7. Proposed Flowchart

12. Conclusion and Future Scope

Developer might copy incorrect code throughout different sections or simply willingly skip updating clones of all codes as a change is present. In our research a review is done on software clone detection to examine hybrid technique based on various parameters. A hybrid technique is proposed that will be designed by merging two

approaches. Quantitative comparison will be providing of our approach with existing techniques. Code cloning is another factor that impacts maintainability of the code while developing software. With additional checks for code cloning by this higher accuracy, better efficiency and performance can achieved for code clone detection than existing approaches.

REFERENCES

[1] J. F. Islam and C. K. Roy, "Bug Replication in Code Clones: An Empirical Study," 2016.

[2] K. Kaur and R. Maini, "A Comprehensive Review of Code Clone Detection Techniques," vol. IV, no. Xii, pp. 43–47, 2015.

[3] R. V. Patil, S. D. Joshi, S. V. Shinde, D. A. Ajagekar, and S. D. Bankar, "Code clone detection using decentralized architecture and code reduction," 2015 Int. Conf. Pervasive Comput. Adv. Commun. Technol. Appl. Soc. ICPC 2015, vol. 0, no. c, 2015.

[4] G. Bansal and R. Tekchandani, "Selecting a set of appropriate metrics for detecting code clones," 2014 7th Int. Conf. Contemp. Comput. IC3 2014, pp. 484–488, 2014.

[5] S. Gupta and P. C. Gupta, "Literature Survey of Clone Detection Techniques," vol. 99, no. 3, pp. 41–44, 2014.

[6] M. Bharti, R. Goyal, and M. Goyal, "Software Cloning and Its Detection Methods 1 1," vol. 8491, no. 2012, pp. 2012–2015, 2014.

[7] B. Priyambadha and S. Rochimah, "Case study on semantic clone detection based on code behavior," Proc. 2014 Int. Conf. Data Softw. Eng. ICODSE 2014, 2014.

[8] M. Tech, C. S. Engg, and M. Gobindgarh, "Hybrid Approach for Efficient Software Clone Detection," vol. 3, no. 2, pp. 406–410, 2013.

[9] P. Prem, "A Review on Code Clone Analysis and Code Clone Detection," vol. 2, no. 12, pp. 43–46, 2013. [10] R. Sivakumar and K. . Kodhai.E, "Code Clones Detection in Websites using Hybrid Approach," Int. J. Comput. Appl., vol. 48, no. 13, pp. 23–27, 2012.

[10] S. U. Rehman, K. Khan, S. Fong, and R. Biuk-Aghai, "An efficient new multi-language clone detection approach from large source code," Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern., pp. 937–940, 2012. 30.

[11] A. Corazza, S. Di Martino, V. Maggio, and G. Scanniello, "A Tree Kernel based approach for clone detection," 2010 IEEE Int. Conf. Softw. Maint., pp. 1–5, 2010.

[12] B. Hummel, E. Juergens, L. Heinemann, and M. Conradt, "Index-based code clone detection: incremental, distributed, scalable," 2010 IEEE Int. Conf. Softw. Maint., pp. 1–9, 2010.

[13] G. M. K. Selim, K. C. Foo, and Y. Zou, "Enhancing source-based clone detection using intermediate representation," Proc. - Work. Conf. Reverse Eng. WCRE, pp. 227–236, 2010.

[14] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Sci. Comput. Program., vol. 74, no. 7, pp. 470–495, 2009.

[15] M. Gabel, L. Jiang, and Z. Su, "Scalable detection of semantic clones," Proc. 13th Int. Conf. Softw. Eng. - ICSE '08, p. 321, 2008.

[16] L. Jiang, G. Misherghi, and Z. Su, "D ECKARD : Scalable and Accurate Tree-based Detection of Code Clones ∗," no. 520320, 2007.

[17] F. Van Rysselberghe and S. Demeyer, "Evaluating Clone Detection Techniques," Evol. Large-Scale Ind. Softw. Appl., no. i, pp. 1–12, 2003.

[18] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," Proceedings. Int. Conf. Softw. Maint. (Cat. No. 98CB36272), pp. 368–377, 1998.

[19] Swarnendu Biswas and Rajiv Mall , " An approach to software engineering", 2009.

[20] Myers, Glen ford. (1979). the Art of Software Testing.

[21] Jovanovich Irena, "Software testing methods and techniques", 2002.