International Journal of Future Generation Communication and Networking
Vol. 13, No. 4, (2020), pp. 1943–1946

# Noval Framework for Code Clone Detection and Management

Neha Saini[a,*], Sukhdip Singh[b]

[a]*Government College Chhachhrauli, Yamunanagar*

[b]*Deenbandhu Chhotu Ram University of Science and Technology, Murthal*

***Abstract***

*Reusing applications is one of the main practises used to save time in the modern age of app creation. Copying and pasting techniques are used to duplicate applications and the mechanism is called the clone code which is known as code cloning. Yet all of this contributes to questions such as upgrading, upgrading time and sustaining. Different clones have been produced in the literature, but it takes too much time and more room to classify the clones. However, in the paper, authors suggest a new approach in which clones can be found that takes less time than current common algorithms to identify clones. In this paper noval framework using Abstract Syntax Trees for code clone detection and management has been proposed which gives better results in terms of precision and recall.*

***Keywords:*** *AST, Code Clone Detection, Bad Smells*

## 1. Introduction

The reuse of the code using copying and pasting operation is very common in the programme design process. As a result, certain parts of the code appear just like clones. Code replication happens commonly in massive information systems [1]. Programmers pick and change a block of code that meets their needs to meet the goals. The use of code cloning is useful in that it contributes to rapid software creation, but it can create significant problems in the production and maintenance of software.

### 1.1. Software Cloning Needs

(1) Time limits: Code cloning constraint is the primary explanation for strict scheduler deadlines [2]. They are meeting these targets using copying and pasting technique.

(2) Vocabulary restriction: when vocabulary does not talk, clones are added. Both real-world challenges have features to contend with.

(3) Developers: Multiple companies calculate the productivity of Developers. The research tester by numbering a total of code lines they evaluated[3]. Copying and pasting the same file to maximise the programme length again.

(4) Large structure of computing: complex software systems are not simple[4]

### 1.2. Code Clones Forms

(1) T1: The only difference in White Spaces is the true duplicate, or observations.

(2) T2: Programming damage syntax is similar to the variables discrepancy, Lyrics, and spaces in steel[5].

(3) T3: code rubble with new, removed or modified statements along with variables, literal facts, conclusions and discrepancies, open spaces and dark spaces[6].

(4) T4: Programming rubble with a different syntax but executing the same features.

## 2. Related Work

The app is composed of a programmer team, Tajima et al. [7] says the code for an assigned section is written by a participant . If the leader implements the whole language, the outcome is software is going to be cloning. This helps in further processing and storage Specifications. Rainer [8] submitted a study focusing, along with its kind, on device replication, duplication and cloning. He spoke about the underlying causes and poor clone results[8]. The study's contribution is details how clones should be

ISSN: 2233-7857 IJFGCN

stopped, current methods tested and clone standards Assessments of the detector[9]. Rattan et al. performed an exhaustive study of the different instruments and clone identification techniques available[10].

## 3. Proposed Framework

```
1. Clones_count=0
2. For each subtree ti:
If Threshold(ti)>=Set_Threshold
Then hash ti to bucket_list
3. For each subtree ti and tj in the same bucket_list
If Compare_both_Tree(ti,tj) > Set_SimilarityThreshold
Then { For each subtree s of ti
  If CheckMember(Clones,s)
  Then RemoveClonePair(Clones,s)
For each subtree s of tj
  If CheckMember(Clones,s)
  Then RemoveClonePair(Clones,s)
AddCloneClass(Clones,ti,tj)
   }
```

*Table 3.1 Algorithm used*

In the proposed framework, algorithm in table 3.1 has been used. In this algorithm first the clone count is set to 0. After that subtrees of the trees that are being compared are matched with each other and clone class is assigned to subtrees.

## 4. Results

| Algorithm | KLOC | Precision | Recall |
|---|---|---|---|
| Dup | 30 | **80** | **80** |
| CCFinder | 30 | **99** | **93** |
| Duploc | 30 | **90** | **86** |
| Our Algorithm | 30 | **97** | **95** |

*Table 4.1 Comparative Analysis*

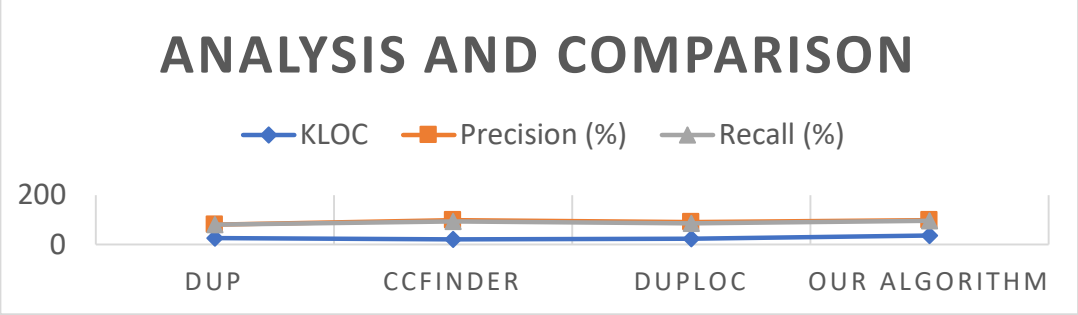


*Figure 4.1 Analysis and Comparison on the basis of Precision and Recall*

## 5. Conclusion

Proposed algorithm is better in terms of precision then previous algorithms like Dup and Duploc. However in terms of recall, it outperforms all three algorithms used for comparison like DUP, Duploc and CCFinder. The complexity of the algorithm is O(n)+O(nlogn). Also it is able to detect clones of all types i.e. type 1, 2, 3 and 4.

## 6. Future Work

In the last few decades, code clone recognition analysis has gone a long way. Finally, some of the related fields that might influence future studies are identified in that area. Several clone detecting tools

ISSN: 2233-7857 IJFGCN

are available. On the other hand, few methods help eliminate or handle copies effectively. The discovery of different approaches to remove dangerous clones by using automatic tool assistance is an important area to analyse the future. The difficulty of how to interpret the vast data generated by clone detection software is also encountered by large-scale clone detection. Another valuable path to the future is graphic and immersive visualisation of the performance to strengthen the human intellect and to provide an actionable perspective. Clone identification is important in many use cases for other programme objects, including templates, bug-reports, relevant records, and binaries. In order to efficiently detect Ransomware and License Breaches, for example, clonesin programme binaries are detected. Applying clone analysis to other software objects is therefore a fruitful direction in the future.

References

[1] Fowler, Martin, and Kent Beck (1999), “Refactoring: improving the design of existing code, Addison-Wesley Professional.

[2] Chanchal Kumar Roy, and James R. Cordy (2007), "A survey on software clone detection research" Queen’s School of Computing TR541 (115): 64-68.

[3] Koschke, Rainer. "Survey of research on software clones(2007), " Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[4] Baker, S.Brenda "On finding duplication and near-duplication in large software systems(1995), " Reverse Engineering, Proceedings of Working Conference IEEE2:86-95.

[5] Rattan, Dhavleesh, Rajesh Bhatia, and Maninder Singh (2013), "Software clone detection: A systematic review" Information and Software Technology55 (7): 1165-1199.

[6]Duala-Ekoko, Ekwa, and Martin P. Robillard(2013), "Clone region descriptors: Representing and tracking duplication in source code." ACM Transactions on Software Engineering and Methodology (TOSEM) 20 (1): 3.

[7] Zhang, Gang, et al(2012), "Cloning practices: Why developers clone and what can be changed." Software Maintenance (ICSM), 2012 28th IEEE International Conference:285-294.

[8] Chatterji, Debarshi, Jeffrey C. Carver, and Nicholas A. Kraft(2013), "Cloning: The need to understand developer intent." Proceedings of the 7th International Workshop on Software Clones. IEEE Press: 14-15.

[9] Komondoor, Raghavan, and Susan Horwitz(2001), "Using slicing to identify duplication in source code." International Static Analysis Symposium. Springer, Berlin, Heidelberg: 40-56.

[10] Ira D.Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant'Anna, and Lorraine Bier. (1998), "Clone detection using abstract syntax trees" Software Maintenance, Proceedings., International Conference on. IEEE:368-377.

[11] Bellon, Stefan, et al(2007), "Comparison and evaluation of clone detection tools." IEEE Transactions on software engineering33(9).

[12] Wagner, Stefan (2013), ”Software product quality control. Berlin”Springer.

[13] Kapser, Cory J., and Michael W. Godfrey (2006), "Supporting the analysis of clones in software systems." Journal of Software: Evolution and Process 18 (2): 61-82.

[14] Kapser, Cory J., and Michael W. Godfrey (2008), “Cloning considered harmful” considered harmful: patterns of cloning in software." Empirical Software Engineering 13(6): 645.

[15] Koschke, Rainer, Raimar Falke, and Pierre Frenzel (2006), "Clone detection using abstract syntax suffix trees." WCRE'06. Working Conference on reverse engineering IEEE(13): 253-262.

ISSN: 2233-7857 IJFGCN

[16] Monden, Akito, Daikai Nakae, Toshihiro Kamiya, Shin-ichi Sato, and Ken-ichi Matsumoto (2002), "Software quality analysis by code clones in industrial legacy software." In Software Metrics, 2002. Proceedings. IEEE Symposium (8):87-94.

[17]Bellon, Stefan, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo (2007), "Comparison and evaluation of clone detection tools." IEEE Transactions on software engineering 33 (9).

[18] M. Mondal, C. K. Roy, and K. A. Schneider (2015), "A comparative study on the bug-proneness of different types of code clones," in Proc. InternationalConference on Software Maintenance and Evolution (ICSME)IEEE: 91–100.

[19] Chatterji, Debarshi, Jeffrey C. Carver, Nicholas A. Kraft, and Jan Harder (2013), "Effects of cloned code on software maintainability: A replicated developer study." Working Conference on Reverse Engineering (WCRE)(20) :112-121.

[20]Jean Mayrand, Claude Leblanc and Ettore Merlo. (1996), "Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics." icsm(96) : 244.

[21]Chanchal K.Roy, James R. Cordy, and Rainer Koschke.. (2009), " Comparison and evaluation of code clone detection techniques and tools: A qualitative approach" Science of computer programming74(7) : 470-495.

[22] Higo, Yoshiki, Ueda Yasushi, Minoru Nishino, and Shinji Kusumoto. (2011), "Incremental code clone detection: A pdg-based approach" Reverse Engineering (WCRE) Working Conference on Reverse Engineering, IEEE18: 3-12

[23] Girija Gupta and Indu Singh (2013), "A Novel Approach Towards Code Clone Detection and Redesigning" International Journal of Advanced Research in Computer Science and Software Engineering3(9): 331-338.

[24] Raheja, Kanika, and Raj Kumar Tekchandani.(2013), "An Efficient Code Clone Detection model on java byte code using hybrid approach"IET : 1-04.

[25] Cheng, Xiao, Hao Zhong, Yuting Chen, Zhenjiang Hu, and Jianjun Zhao (2016), "Rule-directed code clone synchronization" Program Comprehension (ICPC), IEEE International Conference (24) :1-10.

ISSN: 2233-7857 IJFGCN