# Code clones discovery in Asp.net and Java using Hybrid technique.

Manjit Kaur
Assistant Professor
Computer Science and Engineering
Lovely Professional University.
Phagwara, Punjab, India

*Abstract*— **Code clones now a days are very popular and used in many ways such as for Plagiarism checking, String matching,etc. This area of clone discovery is very operational and active research area for most of the researchers.In the proposed paper a hybrid technique is proposed which is a combination of Metric technique for the detection of potential clones and second is Token technique with Suffix array to discover the Actual clones.**

***Keywords—clone, code smells, hybrid, match detection, plagiarism***

## I. INTRODUCTION

People today are very much dependent on internet that they start following the mechanism of copying and pasting the things to complete their tasks. According to software and technology terms, this duplicity achieved by making copying or pasting of things which could lead to lack of originality is referred to as 'cloning' [30]. This copying of codes will lead to copyright infringements of original work of the authorized persons. This has been a question from many years in the mind of researchers who dedicated their research in this field of cloning that whether cloning is a legal or an illegal exercise. Then the answer to this is cloning is not illegal if it has been done with the permission of authorized person. For instance, reusing the code in the software development is an efficient method to reuse the design or requirements; which will save lot of time and cut costs in developing large software products. So to reuse the required things, the owner must be asked about copying his code [30]. In another case also, cloning can not be illegal if the content present on some website or on some journal is open source, but that too leads to the absence of one's originality and creativity. Cloning imparts many cons despite of having many pros, which can be easily justifiable to the fact that "Every rose has its thorns". [30]

## II. RELATED WORK

There has been various proposed and review papers which are advocated by various researchers who worked in this area of cloning and implemented their proposed methodologies using various techniques and tools. [30]

Starting from the year 2006, there was a technique proposed by Rachel Edita Roxas [1] which automatically detects the cloning in students program with the help of the tool named Jplag. In 2007, Stefan Bellon [2] laid out a comparison and evaluation on clone detection tools which says that token and text based techniques works similarly and Marcelo & Baxter's [27] AST tool has higher precision but in parallel, it exhibits higher costs. In year 2008, Cory J. kapser [3] laid emphasis on negative characteristics of clone which are often called code smells. In 2009, a popular review paper stated by James R. Cordy & Chanchal K. Roy [4] was solely based on clone detection techniques and tools. Another technique was given by Tung Thanh Nguyens [5] in the same year whose research work was based on scalable and incremental clone detection with the assistance of Clemanx tool. [30]

In 2010, a technique was proposed by Perumal.A [6]. This technique made use of metrics for extracting similarity in the software clones which had been already detected. In 2011, a technique was proposed by G.Anil Kumar, who used light weight clone detection technique, which is also called hybrid technique as it is the combination of two techniques, metrics and text based along with refactoring support.[30]

In 2012, a technique was proposed by the Saif Ur Rehman & Kamran Khan [7] which constituted of LSC-Miner tool to detect clones. In the same year, another technique was semantic clone detection which was based on JSC Tracker [8] tool and it worked only on java code. Clone detection was further enhanced by Deepak Sethi [9] in the same year. He used a technique of data sets of data mining. In this proposed method,

Solid SDD tool has been used that provides a better visualization of clone detection. Other attempt was made by the Rupinder Kaur & Prabhjot Kaur [12] in the same year which focuses on comparison of two token based techniques known by the name, CC-Finder[21] [24] and PMD to improve performance and efficiency. They concluded that there exists no such tool which detects all the clones efficiently; in fact, each tool has its own strengths and weaknesses which become the factor of its usefulness in detecting clones. [30]

In 2013, Dhavleesh Rattan [13] came up with review paper which was purely related to the various clone detection techniques and methodology which has to be followed while locating clones, pros & cons related to it, etc. In the same year, the other technique was proposed by Kanika Raheja [14]. The technique used here was metric based technique which worked only for Java language.[30]

During 2014, another technique proposed by Harpreet Kaur & Rupinder Kaur [15] was based on metrics. This technique detects clone not only in programming languages but also in web applications. In one of the other clone detection, clones were detected by neural networks and SIMCAD. As data mining seems to be a new emerging area for clone detection another technique was proposed by D. Gayathri Devi [16]. This technique follows an algorithm which detects clones for control structures such as for, while, do statements.[30]

In 2015, one of the latest proposals was given by the Manpreet Kaur and Madan Lal [17], which was a hybrid technique by the combination of the metric based & text based technique. In the present year 2016, Shashank Prabhakar [18] proposed a technique to detect clones which emphasis on detecting Type-1 and Type-2 clones.

In 2017, a technique proposed by Ryota Ami et.al who proposes the Tree based and Token based approach to detect Type1, Type2 and Type3 clones. Another technique was proposed by Ashish N.Runwal et.al in the same year who proposed the Code Clone Detection based on Logical Similarity. Their Proposed system presents an algorithm for clone detection based on comparing parts of abstract syntax tree(AST)of programs and finding semantic coding styles.[34]In the same year Hui-Hui Wei et.al proposed a paper that address the software functional clone detection problem by learning supervised deep features.[35].A technique proposed by Seulbae Kim et called VUDDY, an approach for the scalable detection of vulnerable code clones, which is capable of detecting security vulnerabilities in large software programs efficiently and accurately.[36]One of the Design Code Clone Detection System technique proposed by Jasmandeep Kaur uses Optimal and Intelligence Technique based on Software Engineering.[38]

In 2018, a technique was advocated by Tijana Vislavski et.al who uses LICCA tool which is a cross-language code clone detection tool. Another technique proposed in the same year by Hongfa Xue et.al.They present an novel framework, Clone-Slicer, foridentifying domain-specific binary code clones (e.g., pointer-relatedcode) through program slicing.[37]

Hence, from the literature survey, it can be analyzed and concluded that there are still various active areas in the cloning that can be further worked upon in the future by the new researchers [30] which includes code maintenance, clone removal,etc.

## III. BACKGROUND

*A. Types of clone*

There are four types of clones namely: Exact clone, Renamed or Parameterized clone, Near-miss clone and Semantic clone. These clones are called Type 1 clone, Type 2 clone, Type 3 clone and Type 4 clone respectively.

a) Type 1 clone (Exact clone)

In Type 1 clone, the cloned code is exactly same as that of the original code. In this type of clone, the only difference comes in the form of comments, blank spaces etc; which makes the cloned code different from original code.Text based approach is appropriate to detect this type of clone.

b) Type 2 clone(Renamed/parameterized clone)

In Type 2 clone, the cloned code contains the renaming of variables, literals, etc w.r.t the original code. This kind of clone is detected by Text based technique & Token based technique.

c) Type 3(Near-miss clone)

In Type 3 clone, there is a insertion, deletion and modification of statements in the cloned code w.r.t the original code. Text based, Token based, Hybrid, Tree based and Metric based approach is best for unmasking such clones.

d) Type 4 clone(Semantic clone)

In Type 4 clone, the cloned code is same as that of the original code in terms of functionality or behaviour but differ in the syntax and implementation.

## IV.CLONE DETECTION

### A. *Definition*

It is the process of finding or detecting clones in code. It is used to find clone pairs in programs based on similarity. There are various advantages of finding the clones so as to detect the bugs at the earlier stage, plagiarism detection, etc. [30]

### B. *Steps in clone detection*

There are various steps involved in the detection of clones.[30]

- Pre-processing
- Transform
- Match Detection
- Formatting
- Post Processing
- Aggregation

### C. *Techniques in clone detection*

The various techniques involved in the clone detection are:[30]

- Text Based- This technique compares two code fragments line by line [4][13][14][30]. This technique is only for Type 1 clones. It doesn't consider any renaming of variables and any syntactical or semantically changes. It provides high accuracy. But it is not highly efficient to detect any other kinds of clones.

- Graph Based-This technique uses program dependency graph (PDG). It is good for detecting semantically similar clones. Semantically similar clones are those clones which are syntactically different but show similar behavior or perform same function. In other words, it can detect Type 3 and Type 4 clones efficiently. PDG are directed graph which determines two types of dependencies namely data dependency and control dependency which exists between statements of the source code [30].

- Metric Based-It is a straight forward technique. There are four types of metrics namely class, layout, method and control [4][13][14]. All these types follow a different metrics. Metric based approach is more scalable technique and gives accurate results for large software systems. It contains structural information only. So it is good for finding syntactic clones i.e. Type 1, Type 2 and Type 3 clones [30].

- Token Based-In this technique, there is a formation of lexical tokens [4][13][14]. It is good for detecting Type 1 and Type 2 clones. It gives fast response and is considered to be more efficient as compared to

text based but also gives many false positives. It extracts tokens out of the source code with the help of lexical analysis and based on this token sequence is formed. There is a method called "functor" that maintains the order of tokens [30].

- Tree Based-This technique is based on Abstract Syntax Tree (AST) which is obtained after converting the source fragments into some intermediate form. It is efficient for detecting Type 1, Type 2 and Type 3 clones [4][13][14]. It is a heavy weight technique and requires a sub tree comparison [30].

- Hybrid-This technique is the combination or the union of various other techniques like Tree, Text, Token, Metric and Graph [4][13][14][30] for the detection of code clones.

## V.**PROPOSED WORK**

This paper propose a Hybrid technique with combination of Metric based and Token based technique for multiple languages such as Java and Asp.net language. Metric based technique is used to discover the Potential clones and Token technique has been used where Suffix Array algorithm for Match Detection has been applied on Tokenized sting to calculate Actual clones.

Suffix array computes repeated token subsequences in the given string. Moreover it is a simpler algorithm. The space consumption while constructing the suffix array is five times lesser than other algorithms such as Suffix tree, etc [32].

Given below an example that shows how suffix array works on the string for discover a clones.

| I | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| String | K | E | R | E | L | A |
| SA | 5 | 3 | 1 | 0 | 4 | 2 |
| LCP | 1 | 3 | 5 | 2 | 2 | 0 |

Notations:-
i=Index; String =Sequence of characters where Clone need to be discovered; SA=Array of suffixes arranged in lexicographical order; LCP=Array contains longest common prefix length.

LCP is denoted as LCP[i]=LCP(SA(i-1),SA(i)); where i>0.

Now suffix array is created as :-

0 KERELA
1 ERELA
2 RELA
3 ELA
4 LA
5 A

Sort the suffixes alphabetically →

5 A
3 ELA
1 ERELA
0 KERELA
4 LA
2 RELA

So, suffix array of above string PAPAYA is '5 3 1 0 4 2'

LCP is created as:-

SA of above string is '5 3 1 0 4 2' i.e. (a,ela,erela,kerela,la,rela)

Now,LCP of a,ele=0
LCP of ela,erela=3
LCP of erela ,kerela=5
LCP of kerela ,la=2
LCP of la ,rela=2
LCP of rela,none=0

So, LCP of above string PAPAYA is '1 3 5 2 2 0'

Now clones within the given string will be discovered as follows:-

a) If LCP[i]<LCP[i+1]; Then clones will be discovered in the positions referred by SA[i] and SA[i+1].
b) If LCP[i]=LCP[i+1]; Then clones are sustained at positions SA[i] and SA[i+1].
c) If LCP[i]>LCP[i+1]; Then no clones are discovered.

Given below is the Flowchart which demonstrates the working of proposed algorithm that how the union of Metric approach and Token approach will work out to unmask clones.

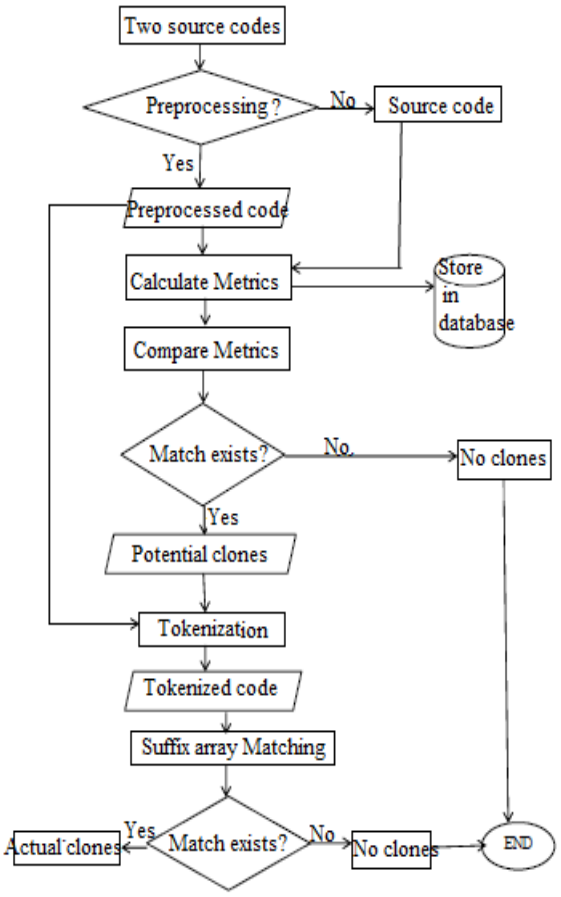


**Figure 1. Flowchart of the Proposed Technique**

The proposed hybrid technique can efficiently detects Type1,Type 2 and Type3 clones with the help of Metric based and Token based Technique which uses Suffix Array matching Algorithm to detect Actual clones. The advantage of using Suffix array algorithm over Suffix Tree Algorithm is, it directly form clone pairs based on the output whereas Suffix Tree requires extra preprocessing step to form clone pair and clone classes[32][33]. Also Suffix array is more memory or space efficient than Suffix Tree. It consumes 5 times less memory than that of suffix tree.[32][33]

## VI. RESULTS ANALYSIS & CONCLUSION

| ***Parameters*** | ***Existing Technique*** | ***Proposed Technique*** |
|---|---|---|
| **Language** | Java | Java+Asp.net |
| **No of Metrics taken** | 14 | 20 |
| **Tool used in Token Approach** | CC-Finder | Tool created in Netbeans 8.1 using Java language |
| **Data structure used in Token approach** | Suffix Tree | Suffix Array |
| **Memory space utilization** | Suffix Tree consumes more memory. Hence less memory efficient. | Suffix array is more memory and space efficient.(Consumes 5 times) |

| | | |
|---|---|---|
| **Time Complexity** | O(n)[33] | O(n)[33] |
| **Types of clones detected** | Type1 and Type2 | Type1,Type2 and Type3 |
| **output** | Require extra preprocessing step to form clone pair and clone classes.[32] | Directly form clone pairs based on output.[32] |

## *References*


[1] Rachel Edita Roxas, "Automation generation of Plagiarism Detection among students Plagiarism", In IEEE Transactions on Software Engineering, September 2006.

[2] Stefan Bellon, Rainer Koschke, Giuliano Antoniol, Jens Krinke, and Ettore Merlo.Comparison and Evaluation of Clone Detection Tools. In IEEE Transactions on Software Engineering, Vol. 33(9): 577-591, September 2007.

[3] C.Kapser and M. Godfrey"Cloning Considered Harmful" Considered Harmfu". In WCRE, pp. 19 -28, 2006.

[4] Chanchal K. Roya, James R. Cordy, Rainer Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of Computer programming, ELSEVIER, pp 470-495, 2009.

[5] Tung Thanh Nguyen, "ClemanX: Incremental Clone Detection Tool for evolving Software", ICSE'09, May 16-24, 2009, Vancouver, Canada 978-1-4244-3494-7/09@ 2009 IEEE.

[6] Kodhai.E, Perumal.A, and Kanmani.S, "Clone Detection using Textual and Metric Analysis to figure out all Types of Clones", Proceedings of the International Joint Journal Conference on Engineering and Technology, pp. 99-103, 2010

[7] Saif Ur Rehman, Kamran Khan, "An Efficient New Multi-Language Clone Detection Approach from Large Source Code," International Conference on Systems, Man, and Cybernetics, IEEE, pp 937-940, 2012.

[8] Rochella Elva,Gary T. Leavens, "A Semantic Clone Detection Tool for Java Code," March 2012.

[9] Deepak Sethi, Manisha Sehrawat, "Detection of code clones using Datasets," IJARCSSE, pp 263-268, July 2012.

[10] Tahira Khatoon, Priyansha Singh, Shikha Shukla, "Abstract Syntax Tree Based Clone Detection for Java Projects," Journal of Engineering, IOSR, pp 45-47, Dec 2012.

[11] Priyanka Bhatta, "HYBRID TECHNIQUE FOR SOFTWARE CODE CLONE DETECTION," International Journal of Computers and Technology, pp 97-102, April 2012.

[12] Rupinder Kaur, H. K, "Evaluation of Token Based Tools On The Basis Of Clone Metrics" International Journal of Advanced Research in Computer Science and Electronics Engineering, 2012.

[13] Dhavleesh Rattan, Rajesh Bhatia, Maninder Singh, "Software Clone Detection: Systematic Review," Information and Software Technology, ELSERVIER, pp 1165-1199, 2013.

[14] Kanika Raheja, Rajkumar Tekchandani, "An Emerging Approach towards Code Clone Detection: Metric Based Approach on Byte Code," IJARCSSE, Vol.3, May 2013.

[15] Rupinder Kaur, Harpreet Kaur, "Clone Detection in Web Application using Clone Metrices" International Journal of Advanced Research in Computer Science and Software Engineering, July 2014.

[16] D.Gayathri Deviet al. "Comparison and evaluation on metrics based approach for detecting code clone" Vol. 2 No. 5 Oct-Nov 2011.

[17] Manpreet Kaur, Madan Lal, "Review on various code clone detection Techniques", Computer Science and Software Department, Punjabi University, May 2015

[18] Shashank Prabhakar, Sonam Gupta "A review on Code Clone Detection and implementation", Computer and Communication Engineering, February 2016

[19] Yogita Sharma "Hybrid Technique for Object Oriented Software Clone Detection", Master Thesis, Computer Science and Engineering Department, Thapar University, June 2011.

[20] Lingxiao Jiang, "DECKARD: Scalable and Accurate Tree-based Detection of Code Clones", pp 2-10.

[21] Mohammed Abdul Bari "Code Cloning: The Analysis, Detection and Removal" International Journal of Computer Applications (0975 – 8887) Volume 20– No.7, April 2011..

[22] Y. Higo, U. Yasushi, M. Nishino, and S. Kusumoto "Incremental code clone detection: A PDG-based approach". In WCRE, pages 3 –12, 2011.

[23] Anna Corazza, Sergio Di Martino, Valerio Maggio, Giuseppe Scanniello, "A Tree Kernel Based Approach for Clone Detection" 26th IEEE International Conference on Software Maintenance (2010).

[24] T. Kamiya, S. Kusumoto, "CCFinder: a multilinguistic token-based code clone detection system for large-scale source code". IEEE Trans. Software. Eng., 28(7):654–670, 2002.

[25] R. Komondoor and S. Horwitz, "Using Slicing to Identify duplication in Source Code", in: Proceedings of the 8th Static Analysis, (2001).

[26] J. Krinke, "Identifying Similar Code with Program dependence Graphs", in Proceedings of the 8th Working conference on Reverse Engineering, (2001).

[27] Ira Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant Anna, "Clone Detection Using Abstract Syntax Trees". In Proceedings of the 14th International Conference on Software Maintenance (ICSM'98), pp. 368-377, Bethesda, Maryland, November 1998.

[28] Baker, B.S, "On finding duplication and near-duplication in large software systems", Proc. of the 2nd IEEE Working Conference on Reverse Engineering, 1995, pp. 86-95.

[29] Sonam Gupta and P. C Gupta, "Literature survey of clone detection techniques." International Journal of Computer Applications (2014):41-44.

[30] M. Kaur, "Review on Software Cloning and Clone Detection," in Interenational Conference on Intelligent Circuits and Systems(ICICS 2016), Phagwara, 2016.

[31] L. P. Z. F. l. M. a. D. S. L. Qing Qing Shi, "A Novel Detection Approach for Statement Clones," IEEE, pp. 27-30, 2013.

[32] H. A. Basit, "Efficient Token Based Clone Detection with Flexible Tokenization," in ACM, 2004.

[33] G. Y. Munina Yusufu, "Efficient Algorithm for Extracting Complete Repeats from Biological Sequences," International Journal of Computer Applications, vol. 128, pp. 33-37, 2015.
[34]Runwal, A. N. (2017). Code Clone Detection based on Logical Similarity. International Journal of Advanced Research in Computer and Communication Engineering, 40-50.
[35]Hui-Hui Wei, M. L. (2017). Supervised Deep Features for Software Functional Clone Detection by ExploitingLexical and Syntactical Information in Source Code. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-1).
[36] Kim, S. (2017). VUDDY: A Scalable Approach for Vulnerable CodeClone Discovery. *IEEE*.
[37] Xue, H. (2018). Clone-Slicer: Detecting Domain Specific Binary Code Clonesthrough Program Slicing. *ACM*.
[38] Kaur, J. (2017). Design Code Clone Detection System uses Optimal and Intelligence Technique based on Software Engineering. International Journal of Advanced Research in Computer Science.