I J C T A, 9(41), 2016, pp. 463-472 ISSN: 0974-5572

Review on Software Cloning and Clone Detection

Manjit Kaur*, Sandeep Kaur and Bhavneesh Sohal*****

ABSTRACT

Today in this modern era of science, technology has developed its roots deep into the world, where most of the things are done with the help of automated tools and techniques to do more work in less time and with great efficiency. This is the case with software industry also. In software industries, a technique called software cloning has come into existence. Software cloning has various broad aspects, out of them; the shadow of light is thrown on one of the aspect called code cloning. In code cloning, some significant quantity of code as desired by the user is taken from some pre-existing code and copied into some another code. In short, it is a kind of copying or pasting of code where some desired code is copied from one source and pasted into another source. The code in which pasting is done is called the replica of original code. In other words, the code which contains the replicated code is called the clone. It leads to the fast development of the software systems. But despite of having so many boons like time saving technique, fast development of software systems, reuse, etc; it has some drawbacks as well like bug propagation.

Keywords: clone, code smells, hybrid, match detection, plagiarism.

1. INTRODUCTION

Modern world is the era of science and technology due to which many new technologies have been introduced at different times. Internet is one of the results of this. Today lot of things are managed online via internet with the help of automated tools, programs, techniques; which surprised people that how things were carried on, when there was no internet. Before this invent, people focused more on reading books, magazines, newspaper, etc to gain knowledge. But now people shift their focus towards the internet to fetch any kind of knowledge. Now Wikipedia, journals and websites are available on internet which provides good and rich amount of knowledge to the people. In a nutshell, internet provides the people with ample of opportunities to do work in more sophisticated manner. It can be said to be an ocean of new technologies, knowledge and many more, to learn many new things out of it. But these all are the one side of coin that how the introduction of internet and advancements related to it, opens new opportunities for people. Now looking at the other side of the coin, these new advancements of internet make people tedious and weary. People now are too much dependent on internet that they start copying and pasting the things to accomplish their tasks instead of learning or grasping them in mind. They are not brainstorming their minds. 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'. 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 is 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

* Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: manjit6454@gmail.com*

** Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: Sandeep.16827@lpu.co.in*

*** Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: Sohal.16042@lpu.co.in*

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. In another case also, cloning can't be illegal if content present on some website or on some journal is for free, 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".

In this paper, second section deals with literature review in the field of cloning since 2006, the third section discusses various advantages and drawbacks with various types of clones currently present. Then in fourth section, various clone detection techniques and their methodologies have been discussed.

2. LITERATURE REVIEW

In literature survey, 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. In this literature review, papers of ten to fifteen years back are taken into consideration.

Starting from the year 2006, there was a technique proposed by Rachel Edita Roxas [1] which automatically detects the cloning in 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 Merlo and Baxter's AST tool has higher precision but in parallel, it exhibits higher costs. They also stated that PDG based tool which was proposed by Krinke does not efficiently detects type 3 clones which are popularly called Near Miss clones. In year 2008, Cory J. kapser [3] laid emphasis on negative characteristics of clone which are often called code smells. They said that these code smells has a bad effect on program design and maintenance. In 2009, a popular review paper stated by James R. Cordy and Chanchal K. Roy [4] was solely based on clone detection techniques and tools. They also gave reviews on clone detection methodology or process involved in the detection of clones. 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, which represents code fragments as sub trees of AST's and detected similar clones on the basis of distance-based clustering problem.

In 2010, many techniques were given by various researchers. Out of those techniques, one 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. After the clone detection, two functions were applied to the detected clones; firstly it could make clusters, where detected clones were grouped together, and then cluster ranking was employed on them by arranging them in ascending order. There was one more review paper that laid out stress on comparison of various plagiarism detection tools like Jplag, Moss, Marble, Gplag, Sim, etc; on the basis of performance and sensitivity. At the end, it was concluded that these tools did not reveal the presence or absence of plagiarism but defined the measure of similarity among the code. 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.

In 2012, a technique was proposed by the Saif Ur Rehman and Kamran Khan [7] which constituted of LSC-Miner tool to detect clones. LSC-Miner is a token based technique and detects clones in multiple languages. It is a good tool which can also work on large software systems. In the same year, another technique which came into play was semantic clone detection which was based on JSC Tracker [8] tool and it worked only on java code. It also used open source database software DSpace and reference management software called JabRef. 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 and this tool was further enhanced with the addition of

graphical user interface. Tahira Khatoon [10] proposed the technique in the same year that uses AST for detecting clones in java language. In this technique, a hash function was introduced to make the partition of AST's into smaller sub trees. Sub trees were also compared to check similarity between them and the sub trees having value, which exceeds the threshold value, were considered as the clones. Another technique given by Priyanka Batta [11] was named as Hybrid technique due to the combination of metric and text based techniques and it could be able to detect the clones in C or C++ language. This tool runs under window platform. This work can be further improved by making the tool to detect clones in multiple languages. Other attempt was made by the Rupinder Kaur and Prabhjot Kaur [12] in the same year which focuses on comparison of two token based techniques known by the name, CC-Finder and PMD to improve performance and efficiency, in which these two techniques were compared via three types of metrics i.e. file metrics, line metrics and clone set metrics. 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.

In 2013, Dhavleesh Rattan [13] came up with review paper which was purely related to cloning background. It was about the various clone detection techniques and methodology which has to be followed while locating clones, pros and cons related to it, etc. Moreover, it was about the clone detection in other key areas like clone analysis, cloning management, 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. The source code was not directly applied on java source code (.java). It was required to be transformed into some intermediate stage where java code has been converted into the java byte code (.class) and then metric based technique was applied to it. These metrics could efficiently discover potential clones which was instrumental in detecting all the possible clones.

During 2014, another technique proposed by Harpreet Kaur and Rupinder Kaur [15] was based on metrics. This technique detects clone not only in programming languages but also in web applications. This also fetched higher level clones called directory clones in java. In one of the other clone detection, clones were detected by neural networks and SIMCAD. This technique was one of the hybrid techniques because it was the combination of two techniques namely text based and data mining based. In data mining technique, K-means clustering was used which proved to be useful and efficient in detecting exact, renamed and near-miss clones. 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. Moving towards the current scenario, there are lots of latest proposals being given by the researchers.

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 and text based technique. Their proposal deals with detection of type-1, type-2 and type-3 clones. Along with it, they also enhanced their technique with the clone removal strategy. For future work, they can work on their tool to make it efficient and capable that it can easily locate type-4 clone also. In the present year 2016, Shashank Prabhakar [18] proposed a technique to detect clones which emphasis on detecting Type-1 and Type-2 clones. Hence, from the literature survey, it can be analyzed that there are various active areas in the cloning that can be further worked upon in the future by the new researchers.

3. BACKGROUND

3.1. Reasons of cloning

- Reuse-It is an efficient method to reuse the design code and its requirements. Hence, it saves time and reduces cost.
- Templates- Some programming paradigms have encouraged the use of templates.

- Coincidence- Sometimes different developers unintentionally write the same piece of codes without knowing other's code.
- Large codes- Fear of writing large codes also encourages programmers to copy the code.
- Complexity of the system- Difficulty in understanding large systems also promotes copying the existing functionality and logic.

3.2. Advantages of cloning

- Lack of Knowledge about language- Some programmers doing well at one language while not so good in others. This is due to the lack of knowledge of the programming language.
- On-time software development- In software development, scheduling of tasks to each developer has been assigned to complete the work on time or to meet the deadlines.
- Fast method- It is considered as a fast method for developing software systems.

3.3. Disadvantages of cloning

- Effect on maintainability- Cloning has an adverse effect on maintainability as it invites more maintenance cost.
- Bug -propagation- It also lead to bug propagation from original code to the copied one.
- Effect software evolution- It becomes hurdle for better evolution of software systems as it has bad impact on designing and many other areas of software.
- Lack of originality- Often developers copy some part of code from some websites and paste them in their source code. This process of copying and pasting results in loss of originality.

3.4. Types of clone

To detect clones, it is necessary to be aware of different kinds of clones. There are four types of clones namely: Exact clone, Renamed or Parameterized clone, Near-miss clone and Semantic clone. These clones are also called Type 1, Type 2, Type 3 and Type 4 clones respectively. These clones are different according to different specialties and these specialties are discussed as under:

3.4.1. Type 1 clone(Exact clone)

Exact clones are the clones which look like an original code. These clones can be easily detected with the help of simplest clone detection technique like text based, token based, etc. The difference comes only in the blank spaces or in the comments. These can be easily detected by text based techniques and tools like SDD, KMP algorithm for string matching, etc.

Table 1
Exact clone

CODE	CLONE

int add(int no[],int x) {
int g=0; //add
for(int t=0; t<x; t++) {
g=g+ no [t];}
return g;
}
	int add(int no[],int x) {
	int g=0; //addition
	for(int t=0; t<x; t++) {
	g=g+ no [t];}
	return g;
	}

3.4.2. Type 2 clone(Renamed/parameterized clone)

Renamed clones are the clones where variations come in the name of literals, keywords, variables, etc. These clones can be detected by techniques called token based, metric based , etc. Various tools are also implemented by developers to detect these types of clones. These tools are CLAN, MCD-Finder, Columbus, etc.

Table 2
Renamed clone

CODE	CLONE

int add(int no[],int x) {
int g=0; //add
for(int r=0; r<x; r++) {
g=g+ no [r];
}
return g;
}
	int addition(int s[],int x) {
	int q=0; //sum
	for(int p=0; p<x; p++) {
	q=q+ s [p];
	}
	return q;
	}

3.4.3. Type 3 (Near-miss clone)

In these types of clones, changes persist in code in the form of addition, deletion and modification of statements. These clones can be detected by the techniques called tree based and tools called Clone Digger, CloneDr, etc. These clones can be easily detected by tree based techniques, where sub trees of AST are being compared with each other.

Table 3
Near-miss clone

CODE	CLONE

int addition(int no[], int n)
{
int sum=0; //sum
for(int i=0; i<n; i++) {
{
sum=sum+ no [i];
}
return sum;
}
	int sum(int a[], int n)
	{
	int t=0; //add
	for(int y=0; y<n) {
	{
	t=t+ a [y];
	y++;
	}
	return t;
	}

3.4.4. Type 4 clone(Semantic clone)

In these clones, function or behavior of the clone remains same but syntax or coding of program is different. These types of clones are detected by using graph based techniques and tools like Scorpio, Duplix, etc.

Table 4
Semantic clone

CODE	CLONE

int add(int no[],int n){
int d=0;
for(int e=0; e<n; e++){
d = d+ no [e];
}
return d;
}
	int add(int no[],int q){
	if(q==1)
	return no[q-1];
	else
	return no[q-1]+add[no,q-1];
	}

4. CLONE DETECTION

4.1. 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 clones so as to detect the bugs at the earlier stage.

There are various steps that are involved in clone detection and many techniques are used to detect the clones efficiently with the help of tools.

4.2. Advantages of clone detection

- Software analysis- It is very useful for software analysis and understanding of software evolution.
- Bug Detection- It finds bugs in the program so that they are not propagated from one program to another.
- Understand ability- It enhances program understand ability and reduces program size.
- Plagiarism Detection- It is having biggest advantage in detecting plagiarism in order to protect copyrighted content from being copied.

4.3. Steps in clone detection

I. Pre-processing
II. Transform
III. Match Detection
IV. Formatting
V. Post Processing
VI. Aggregation

4.3.1. Pre-processing

- Remove unnecessary parts- All the source code which seems to be irrelevant should be discarded in the comparison phase [4]. For e.g., If the tool is not language independent, then different languages needed to be separated from code like separating sql from java code.
- Determine source units- The remaining code obtained after removing the uninteresting code, will be bifurcated into a set of disjoint fragments and these fragments are known by the name 'source units' [4].
- Determine comparison units/granularity-Based on the comparison technique used by the tool, these source units are further divided into much smaller units [4].

4.3.2. Transform

It converts source units which are needed for comparison, into some intermediate state. All the techniques [4], except text based, require a transformation of the source units. This transformation is also referred to as 'extraction' according reverse engineering community. This transformation can be achieved in two ways: extraction and normalization.

1. *Extraction*- It is further bifurcated into 3 sub-categories namely tokenization, parsing, control and data flow analysis [4].

   - Tokenization- In this approach, source units are converted into tokens based on lexical protocols or procedures and these tokens are arranged in token sequences, after the removal of blank spaces and comments, for comparison.

- Parsing- Here, entire source code is parsed to generate an AST (Abstract Syntax Tree) and then source units from AST's which are needed are shown in the form of sub trees. To figure out clones, these sub trees need to be compared.
- Control and data flow analysis- In this approach, PDG (Program Dependency Graph) generated tools create PDG graphs in which nodes represent statements whereas edges represent data and control dependency. To lay out a comparison, sub graphs of PDG's are compared.

2. *Normalization*- This is an optional step to eliminate differences based on comments, whitespaces, etc. This can be achieved in many ways like by normalizing the identifiers where all identifiers in source code are replaced by the single identifier, pretty-printing, etc [4].

4.3.3. Match Detection

In this phase, transformed units obtained from the transformation phase are passed into some comparison algorithm and then compared to find a proper match [4]. The output contains a list of matches in the transformed code which represents the clone relations in the form of clone pairs, clone classes and clone family. Certain comparison approaches include hashing, suffix trees, etc.

4.3.4. Formatting

In match detection, clone pair list is obtained for transformed code but in this phase, the list is further converted into another pair of list that matches with the original code base [4].

4.3.5. Post Processing

In this phase, clones are filtered or ranked using manual analysis or automated heuristics [4]. In manual analysis, false positive clones are filtered out by a human expert. Automated heuristics is based on length, diversity, frequency and other characteristics of clones in order to rank or filter out clone candidates automatically.

4.3.6. Aggregation

This is the last step of the clone detection process. It refers to proper analysis and data contraction. The detected clone pairs are combined to form clone classes and clone family [4].

4.4. Techniques in clone detection

- Text Based
- Graph Based
- Metric Based
- Token Based
- Tree Based
- Hybrid

I. *Text Based*: This technique compares two code fragments line by line [4] [13] [14]. 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. Many researchers come up with new tools and technologies like Johnson et al. proposed a fingerprinting technique for detecting text based clone fragments. Another tool is DUPLOC which is devised by Ducasse et al. It is a language independent tool which requires no parsing of source code i.e. it is directly imposed on source

code to detect clones. Similar Data Detection (SDD) tool detects clones in systems of large size. Similarly, Barbour et al. stated a technique based on an algorithm popularly known as Knuth-Morris-Pratt (KMP) which compares strings to detect clones.

II. *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. Tools under this technique are Duplix, PDG-Dup, Scorpio, etc. Duplix tool is proposed by Krinke et al. It finds maximum similar sub graphs from the transformed source code. PDG-Dup traces the isomorphic sub graph with the help of program slicing and it is proposed by Komodor et al. Scorpio is stated by Higo and Kusumoto et al. In this tool, two way slicing is introduced i.e. forward slicing and backward slicing. If clone is not detected in the forward slicing, it can be detected in backward slicing. There is another tool which is proposed by Liu et al. It is a PDG based plagiarism detection tool and algorithm on the basis of program dependence.

III. *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. CLAN is a tool advocated by Mayrand et al. In this technique, AST of a source code can be collected to compare metrics based on it. There is another similar technique introduced by Kontogiannis et al. that applies dynamic programming on metrics. MCD-FINDER [14] is proposed by perumal et al. In this technique, Fingerprinting approach is used for clone detection in source code. Dagenais et al. proposes a technique which is applied on textual representative of source code.

IV. *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. Tool called CP-MINER is based on data mining approach. It makes use of frequent item set mining which is helpful in bugs and structural clone detection. Likewise, CC-FINDER [12] [20] tool devised by Kamiya et al. is used to find identical subsequences with suffix matching algorithm. It detects clones in languages like C, C++, Java, COBOL, etc. Another tool FC-FINDER requires hashing to detect file clones. Similarly, a tool which is known as LSC-MINER detects clones in large source codes. Basically, it is a multilinguistic tool that is used to detect clones in more than one language. It is implemented in VisualBasic.Net.

V. *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. Under this technique, various tools and methods are proposed by the researchers to detect clones that fall under type 1, 2 and 3 categories. A popular tool called CLONEDR is used to fetch near miss clones. It uses two most popular approaches namely hashing and dynamic programming. This tool comes in two variations: Simscan and Ccdiml. Simscan fetches parsed source code and performs sub tree comparison on it whereas Ccdiml transforms the source code into some intermediate form. CLONEDIGGER is another popular tool which is language independent and merely based on XML which is language independent, extensible and machine understandable. This tool requires XML representation of source code. It also makes use of anti-unification approach to put similar kind of clones in one cluster based on measured distance. Likewise, a tool called DECKARD [19] is more scalable and accurate tool than any other tool. It is a language independent tool

which is proposed by Ling Xiao Jiang.

VI. *Hybrid*: This technique is the combination of various other techniques like tree, text, token, metric, graph [4][13][14]. Under this technique, a tool called Seclone is proposed by Keivantoo et al. This tool consists of four stages: preprocessing, indexing, searching and post processing. Another tool is HCDETECTOR, which merges PDG based technique and metrics based technique. It only works for java language and does its execution on java byte code (.class) which is the intermediate stage of java source code (.java), rather than on original code itself.

5. CONCLUSION

From the study of various research papers, it can be concluded that cloning is in great demand today apart from its various shortcomings. It has proven to be an advantageous process in fast development of the software systems to meet the deadlines or to complete the work on time, etc. It is considered as a great boon to industries. Also, various tools and techniques have been proposed to detect the clones, wherever required, to overcome the various pitfalls released by cloning like bug propagation, maintenance costs, etc. Further detection process comprises of different levels such as preprocessing, transformation, match detection and so on. These techniques and tools can detect various types of clones according to their efficiency and ability. Furthermore, research in this field is still in continuation for introduction of new highly efficient and applicable tools or techniques that will meet up with the requirement of detecting all types of clones because as discussed earlier, there are still many techniques and tools which are not able to detect all types of clones.

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, "Cleman X: 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 Devi et 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] Lingxiao Jiang, “DECKARD: Scalable and Accurate Tree-based Detection of Code Clones”, pp. 2-10.

[20] 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.