ISSN No. 0976-5697

Volume 5, No. 5, May-June 2014

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Code Cloning: Types and Detection Techniques

Noble Khatra
Computer Science and Engineering Department
G.Z.S P.T.U Campus
Bathinda, India

Shaveta Rani
Computer Science and Engineering Department
G.Z.S P.T.U Campus
Bathinda, India

Paramjeet Singh
Computer Science and Engineering Department
G.Z.S P.T.U Campus
Bathinda, India

Abstract: Code clones presence is being recognised as emerging cause of concern in software industry. The presence of code clones makes the software maintenance extremely difficult. There exist different varieties of code clone which need to be tackled at the earliest in order to provide a smooth functioning to the industry. Code clones identification thus becomes extremely necessary in order to avoid the problems caused by them. This paper aims to explain about the various approaches to detect code clones as well as different types of code clones.

Keywords: Code cloning, types of code clone, code clone detection techniques.

I. INTRODUCTION

With the advancement of software industry, the IT sector is becoming more prone to the phenomenon of software piracy. Out of the various factors, one factor that results in software piracy is code clones that are becoming increasingly rampant causing harm to the IT world. Code clones cause a serious threat to the security and legitimate rights of the customers and the IT companies. Thus, there arises a serious need to detect and check the various types of code clones [7].

Code clones are the sections of very similar or identical code that are obtained by reusing of code fragments by copying and pasting with or without any major or minor adaptations[1]. Due to the reason of copy- and- paste programming code clones have resulted in generating various issues as listed below:

- Hampers code reusability and maintainability.
  Long repeated sections of code are generated that differ in only a few lines or characters.
- Hides what the specific purpose of each code section is.
- Increases redundancy which has to be avoided.
- Increases the maintenance costs.

In a software system it has been stated that around 5% to 10% code are cloned and around 60% of the efforts of the organization is wasted in maintaining the existing software [3, 4 and 16]. Due to this rapid increase in the code clones and the resulting maintenance problems, more and more focus is being shifted to the detection of the various types of code clone [2]. Paper [14] states the following:

Language dependency is a big obstacle when it comes to the practical applicability of duplication detection. We have thus chosen to employ a technique that is as simple as possible and prove that it is effective in finding duplication

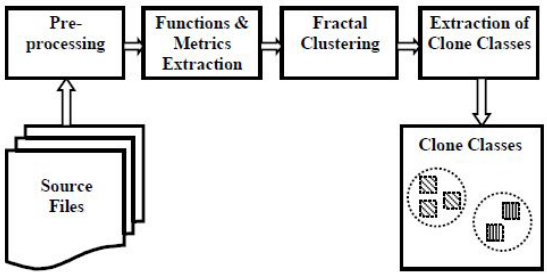

Figure 1: The clone detection process [12].

II. TYPES OF CODE CLONE

[8] States that there exist four types of code clones as described below:

A. *Type 1*

It is also called exact copy clone. In type 1 some variations exist in the form of change of comments or in white spaces.

Table I: Example of Type 1

Code 1	Code 2

//Function with multiple equations
void equation()
{
int a,b,c,d,e;
c= a+b;
e=c-d;
}
	//Function with multiple equations
	void equation()
	{
	int a,b,c,d,e;
	c= a+b;
	e=c-d;
	}

B. Type 2

It is syntactical same copy. In type 2 literals are changed e.g. name of variables and name of functions are changed. It is difficult to detect as compared to type 1.

Table II: Example of Type 2

Code 1	Code 2

//Function with multiple equations
void equation()
{
int a,b,c,d,e;
c= a+b;
e=c-d;
}
	//Multiple equation function
	void muleqa()
	{
	int aa,bb,cc,dd,ee;
	cc=aa+bb;
	ee=cc-dd;
	}

C. Type 3

It is code clone in which lines are added or deleted and lines are interchanged.

Table III: Example of Type 3

Code 1	Code 2

//Function with multiple equations
void equation()
{
int a,b,c,d,e;
c= a+b;
e=c-d;
}
	//Multiple equation function
	void muleqa()
	{
	int aa,bb,cc,dd,ee;
	ee=bb-dd;
	cc=aa+ee;
	}

D. Type 4

It is a code clone which is not created intentionally. These types of clones are created un-knowingly the presence of similar code. These are very difficult to detect.

III. RELATED WORK

Sarkar, M et al [13] presents a hybrid clone-detection technique, consisting of metrics-based, PDG-based and AST-based clone detection, in order to make the clone detection process more reliable and robust. They focus on the use of clone detection techniques for resource requirement prediction of jobs running in a large and dynamic distributed system.

Murakami, H et al [9] propose a new detection method that is free from the influence of the presence of repeated instructions. In the proposed method there is transformation of every of repeated instructions into a special form, and then using a suffix array algorithm it detects code clones.

Chanchal K. Roy and James R. Cordy [3] discussed the different techniques of software cloning. They first discuss various techniques and then compared these techniques with scenario based evaluation.

Yoshiki Higo, Yasushi Ueda, [4] discussed PDG approach of code clone detection. They developed a prototype tool, and applied it to against open source software. The experiment showed that the proposed method could obtain code clones within a short timeframe and its detection result was quite similar to the detection result of an existing PDG-based detection tool.

Hummel et al. [1] proposed an index based code clone detection methodology. Their method firstly replaces user defined identifiers with special tokens in every line of the source code. Then, hash values are calculated from them. Next, the method stores their hash values, their line numbers, and their files names into the database. By using the database, lines that are duplicated with specified lines can be instantly obtained. Multiple lines duplication can be easily constructed by combining single-line duplication stored in the database.

IV. VARIOUS METHODS TO DETECT CODE CLONING

A. Text Based

It requires little or no transformation or normalization. In this approach code slices are considered as sequences of strings and then these are compared with each other in order to find the same strings [6]. This approach can detect Type-1 code clone but cannot detect the structural type of clones having different coding but same logic [8].

B. Token Based

This approach uses parser or lexer for the transformation of source code into a sequence of tokens [5]. Then the scanning of these sequences of tokens is done to find the same token sequences [13, 15]. The original code slices are that are represented by the token sequences will then be returned as clones. This approach though more efficient than text based approach if there exists blank spaces and comments but its accuracy level is not satisfactory as various false positive clones will be introduced in the code while conversion of source code in the token sequence[8]. Type 2 code clones can be detected using this method.

Table IV: Simple and Normalized Code for token based

Simple Code	Normalized Code

void equation()
{
inta,b,c,d,e;
c= a+b;
e=c-d;
}
	void $id()
	{
	int $id,$id,$id,$id,$id;
	$id= $id+$id;
	$id=$id-$id;
	}

C. *Abstract Syntax Tree Based*

In AST the clones are searched by searching for similar sub-trees. The suspected clones returned are the original code slices represented by the sub- trees. The level of accuracy is considered good in this approach but it results in unstable scalability as it depends on the algorithm that is being used to build and compare of the trees [9].

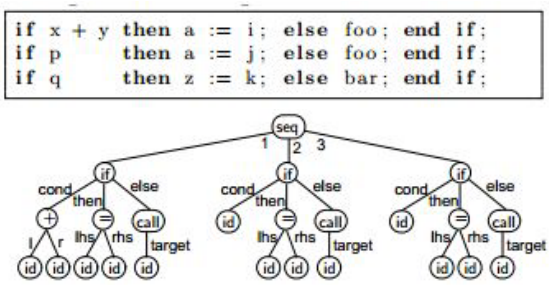

Figure 2: Sample code and abstract syntax tree of code

D. Program Dependency Graph Based

This approach lays emphasis on data dependencies and control flow. Isomorphic sub-graph matching algorithm is applied once the PDG is obtained from the source code [11]. This helps in the finding of the clones. PDG- based detection approach is very effective as it can detect non- contiguous code clones. But it is a cumbersome and costly process to obtain PDG for large software.

```
void bar() {
  int j = 1;
  int i = 0;
  while (j < 10)
    j++;
  printf("%d", i);
  printf("%d", j);
}
```

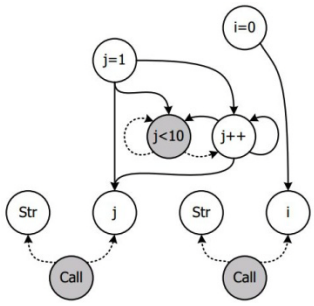

Figure 3: Sample code and dependency graph

E. Metric Based

Here metrics are used to measure clones in software after the calculation of metrics from source code. Metric based approach parses the source code to its AST/PDG representation for the calculation of metric [9]. For the purpose of calculation of metrics from source code various tools like Columbus, Source monitor are available [8]. Metric based approach has high accuracy and scalability level. Some common fields used in metric based detection:

- Number of declaration statements (Decl.)
- Number of executable statements (Stmt.)
- Number of conditional statements (Cond.)
- Number of looping statements (Loop)
- Maximum nesting level of control constructs (Nest)
- Number of return statements (Ret.)
- Number of parameters (Param.)
- Number of called functions (Call)

F. Line Based

In line based techniques code is matched for each line. In type 3 code clone the lines of code are interchanged or lines are added or deleted. So, it is necessary to have some way to check code line by line rather than complete matching of code. In line based technique each line of first code is matched to each line of other code. Line Based technique has high accuracy [10, 13].

V. CONCLUSION

In this paper, effort has been put in to explain about code cloning, disadvantages of presence of code clone, types of code clones in which type 1,type 2, type 3 and type 4 code clones are defined. Also the various types of code clone detection techniques are explained.

In future, some new technique can be proposed to detect code clone with high accuracy or some techniques can be mixed to create a hybrid approach of code clone detection.

VI. REFERENCES

[1] B. Hummel, E. Juergens, et al. "Index-Based Code Clone Detection: Incremental, Distributed, Scalable", *In Proc. of the 26th IEEE International Conference on Software Maintenance*, pp 1–9. 2010.

[2] B. Stefan, K. Rainer, et al., " Comparison and Evaluation of Clone Detection Tools", *IEEE Transactions on Software Engineering*, pp.577–591, vol.33, no.9, 2007.

[3] C.K. Roy and J.R. Cordy. "A Survey on Software Clone Detection Research. School of Computing TR", *Queen's University*, pp 1-115 . 2007.

[4] H. Yoshiki, U. Yasushi, et al., "Incremental Code Clone Detection: A PDG-based Approach", IEEE18th *Working Conference on Reverse Engineering*. pp 3 – 12, 2011.

[5] I.Mai,O. Shunsuke and N. Takuo, "Token-based Code Clone Detection Technique in a Student's Programming Exercise" , *2012 Seventh International Conference on broadband, Wireless Computing*, Communication and Applications, Victoria, BC, pp. 650-655. 2012.

[6] J. Krinke. , "Is Cloned Code more stable than Non- Cloned Code?", In Proc. of the 8th IEEE International Working Conference on Source Code Analysis and Manipulation, pp. 57–66, Oct. 2008.

[7] J. Krinke, "Identifying Similar Code with Program Dependence Graphs", *Working Conference on Reverse Engineering*. pp. 301-309, 2001.

[8] K. Rainer. Raimar, F. Pierre, "Clone Detection Using Abstract Syntax Suffix Trees", *Working Conference on Reverse Engineering*, pp 253-262. 2006.

[9] M. Hiroaki, H. Keisuke et al. , "Folding Repeated Instructions for Improving Token-based Code Clone Detection",*2012 IEEE 12th International Workinq Conference on Source Code Analysis and Manipulation*, Trento, pp. 64 – 73. 2012.

[10] M. Kazuaki, "An Extended Line-Based Approach to Detect Code Clones Using Syntactic and Lexical Information", *IEEESeventh International Conference on Information Technology*, Vegas, NV, pp.1237 – 1240, 2010.

[11] Nguyen H.A., Nguyen T.T. et.al,"Clone Management for Evolving Software", *IEEE transactions on software engineering*, vol 38 no 5, pp 1008-1026. 2012.

[12] Salwa K. Abd-El-Hafiz ,A Metrics-Based Data Mining Approach for Software Clone Detection,2012 IEEE 36th International Conference on Computer Software and Applications, pp35-41,Izmir. 2012.

[13] Sarkar, M.; Chudamani, S. ; Roy, S. ; Mukherjee, N. et al, "A hybrid clone detection technique for estimation of resource requirements of a job", *Third International Conference on Advanced Computing & Communication Technologies*, Rohtak, pp. 174-181. 2013.

[14] S. Ducasse, M. Rieger, and Serge Demeyer, *A Language Independent Approach for Detecting Duplicated Code*, 15th IEEE International Conference on Software Maintenance, pp.109–118.1999.

[15] T. Kamiya, S. Kusumoto, and K. Inoue. "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code", *IEEE Transactions on Software Engineering, vol*. 28, no.7 pp 654-670. 2002.

[16] Zibran M.F. and Roy C. K.,"Conflict-aware optimal scheduling of prioritised code clone refactoring", IETSoftw, vol 7 no 3, pp 167-186,2013.

