Technium Vol. 4, Issue 3 pp.28-36 (2022)
ISSN: 2668-778X
www.techniumscience.com

Survey on Software Code Clone Detection

Yasir Mohammed Khazaal [1], Asma'a Y. Hammo[2]

1*Software, Computer Science and Mathematics, Mosul, Mosul, Iraq

2*Software, Computer Science and Mathematics, Mosul, Mosul,

1* yasirmk@uomosul.edu.iq 2* asmahammo@uomosul.edu.iq

Abstract. Software code clones refer to the same part of code that appears in different places. Software clones (with or without edits) may occur during the software development for the same requirements. This copy and paste may be done on different levels (file, class, and method). Despite this process reduces the time and effort required to build the software but on the other hand, it may increase the cost and effort required to maintain the software. Also, may lead to bugs propagation when the bug occurs in the original copy of code. For this reason, many tools for code clone detection are implemented in the last 20 years to detect different kinds of clones (kind1, kind 2, kind 3, and kind 4) using different methods for different programming languages (Java, C, C++…etc).

This paper explains: what are software clones, their kinds, and the methods used to detect them. It contains also a list of researches in this field till Jan 2022.

Keywords. software clone, clone kinds, clone types, clone detection techniques or methods.

1. Introduction

Software code (source code) is s sequence of code, with remarks or without, written using a computerized high-level programming language. The source code of a program is typically written by a computer programmer and is used to execute the actions that a computer will perform. It is translated to assembly or binary code by a compiler. The binary machine code can be executed by the CPU of the computer. Software code clones refer to the same part of code that appears in different places. If cloning occurs within the same project, such clone pairs are called intra-project and may indicate the necessity to refactor the project. Or it may accrue between different projects then is called Inter-project clones [18].

2. Kinds of clone

2.1 kind 1 (Exact clone)

In this kind of clone, the two original codes are identical except for the blank spaces and comments. They can be easily detected by text-based or token-based methods. Tools based on algorithms like

KMP (Knuth Morris Pratt) and SDD (Similar Data Detection) algorithm can detect this kind by string matching [1,12,20].

2.2 Kind 2 (Renamed/Parameterized)
Here the differences are only in name of variables and keywords. The use of "token-based" or "metric-based" methods can detect these kinds of clones in a code. Tools such as CLAN, Columbus, and MCD-finder can detect this kind of clone. [1,12,20].

2.3 Kind 3 (Gapped clone)
In this kind, statements in the source code are changed (by adding or deleting). tools like "ClonedDr" and "Clone Digger" based on the "Abstract syntax tree" method can be used to detect "near-miss clones" [1,12,20].

2.4 Kind 4 (Semantic or logical clone)
In this kind, the program statements are different while the action or function of the clone remains the same in "semantic clones". Tools like "Duplex", and "Scorpio" use the "program dependency graph" method can detect this kind of clone. [1,12,20].

3. Clone detection Methods
To detect the previously mentioned kinds of clones, there are many methods. They are:

3.1 Text-based method
the goal of this method is to find the clone pair. Clone pair is obtained in the following steps: First do the Pre-processing (remove not-important parts such as Comments, Blank Spaces, …. etc.). Then the obtained code is Compared character by character or line (sequence of characters) by line with the source code. The similarity is called the "clone pair" [1,20].

3.2 Token-based method
It is based on generating tokens of the source code. The lexical parser is used usually for generating tokens. For example, the ASTVisitor tool. Then tokens are indexed. The similarity indexes among token sequences indicate there is a clone [1,20].

3.3 Metric-based method
In this method, while doing the pre-processing of the code, various metrics are calculated. These metrics are matched with each other. When finding the similarity in these criteria, the code segments are mentioned as a clone or a non-clone pair. different criteria like "fan-out", "MaCabe cyclomatic code" the complexity of source code, and the attribution of the number of the input variables to the output variables can be used in this method. The important note about this method is that it has low portability but high scalability. [1,20].

3.4 Abstract syntax tree (AST) -based method
To detect the clone detection using this method, the following steps should be followed:
First of all, an AST of the Source code is made by using available tools, second, the developer should use matching algorithms to find similar subtrees. In the case found, the clone pair is mentioned. Otherwise, it's mentioned as a non-clone pair. [1,20].

3.5 Program Dependency Graph (PDG) -based method
This is the best method because it can detect logical (Kind 4) clones in the source code efficiently. Data flow and function of the source code are used in the detection process. They can be obtained by the program dependency graph [1,20].

4. Steps of the clone detection process
To detect the clone in a source code the following steps should be followed:

4.1 Pre-processing
The input of this phase is the source code. This phase divides the code into two parts: the important part that is useful in clone detection and the not-important parts like Comments, Blank Spaces, …. etc. It also specifies the source unit where the comparison is achieved for clone detection in the source code. The result of this stage is called pre-processed code. [1,20]

4.2 Transformation
The input of this step is the pre-processed code. Then it's converted into different forms such as "token form", "abstract syntax tree", "program dependency graph", etc. which make the comparison easier to detect the code clone. [1,20]

4.3 Match Detection
The input of this step is the transformed code. There are many methods to detect the similarity such as: "similarity index", "string matching" algorithm, "subgraphs matching". The result of this step is the clone and non-clone pairs. [1,20]

4.4 Formatting
This step comes after matching. It specifies the location of clone pairs and classes in the source code. [1,20]

4.5 post-processing (filtering process)
This phase removes all non-clone pairs. So, the clone management now can be performed on the code clone pairs. [1,20]
Figure 1- shows the Steps of the clone detection process [1].

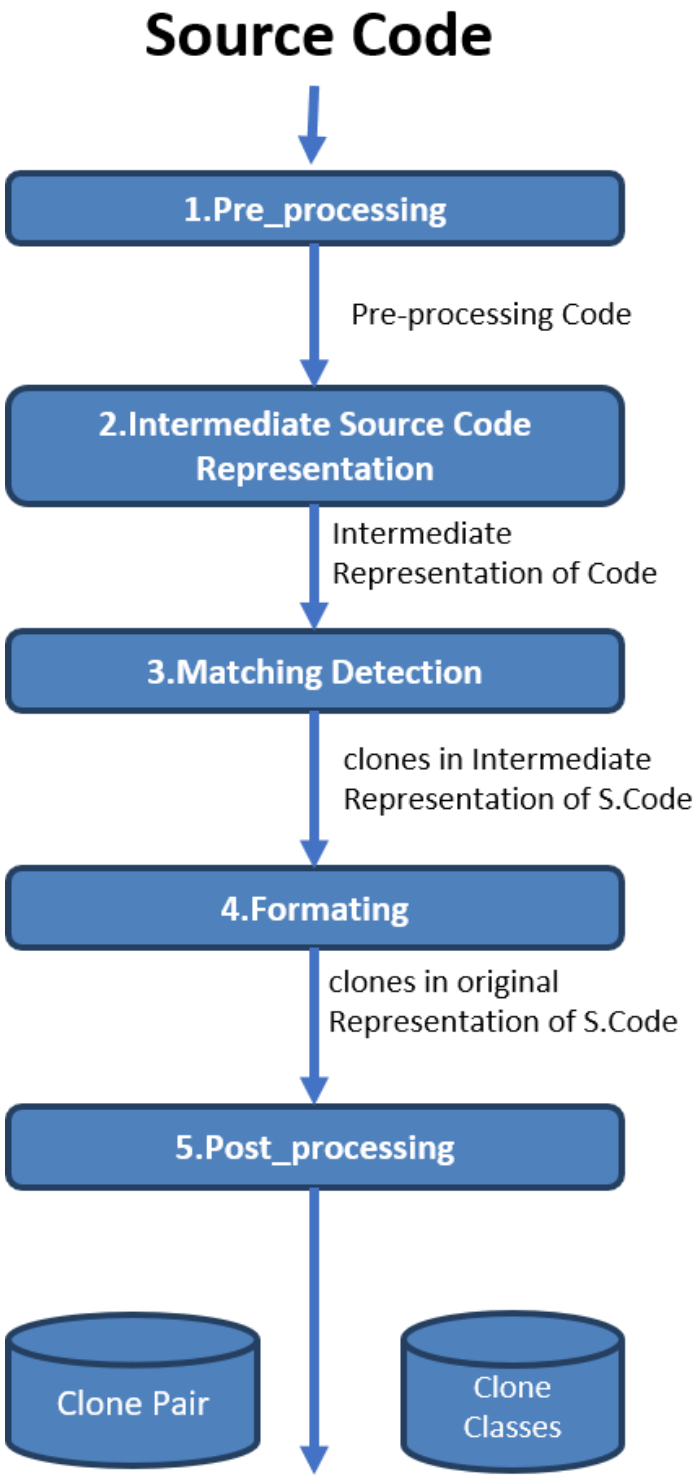

Figure 1- Steps of clone detection process [1]

5. Related work

Table 1 gives the research that are done in the last 10 years about clone detection in Software.

Table 1-A survey of software clone detection using different methods and kinds

Name of researcher	Year	Name of tool	kinds of clone	Methods	Language
"Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue"[21]	2002	"CCFinder"	Kind-1	Token_based	C, C++, Java, COBOL
"Kodhai. E, Perumal. A, and Kanmani.S"[22]	2010	NA	Kind -1, Kind -2, Kind -3	Text-based and Matric-based	Java language
"Priyanka Batta, Miss Himanshi"[23]	2012	NA	Kind-1	Hybrid(metric approach with text-based (line of code))	C/C++ language
"Philippe Charland Benjamin C. M. Fung Mohammad Reza Farhad"[24]	2014	NA	Kind -1, Kind -2, Kind -3	Token_based	ASSEMBLY language
"Abdullah Sheneamer Jugal Kalita"[25]	2015	NA	Kind -1, Kind -2, Kind -3	Token-based Metrics'-based	NA
"Sergej Chodarev, Emília Pietrikova, Jan Kollar"[26]	2015	NA	Kind -1, Kind -2	AST-based	NA
"Sudhamani M, Lalitha Rangarajan"[2]	2016	NA	Kind -1, Kind -2, Kind -3, Kind -4	Metric-based	Source Code
"Yusuke Yuki, Yoshiki Higo, Shinji Kusumoto"[27]	2017	NA	Kind -1, Kind -2, Kind -3	Coarse detection	Java language
"Ashish N.Runwal Mr. Akash-D.Waghmare"[3]	2017	NA	Kind-4	abstract syntax tree (AST)	C++ and Java languge
"Roopam, Gurpreet Singh"[4]	2017	NA	Kind -1, Kind -2, Kind -3	Hybrid(metric-based and PDG-based)	Java language
"Liuqing Li, He Feng, Wenjie Zhuang, Na Meng, Barbara Ryder"[5]	2017	"CCLearner"	Kind -1, Kind -2, Kind -3, Kind -4	Token-based ANTLR and Eclipse STParser	C language

"Yuichi Semura, Norihiro Yoshida, Eunjong Choi, Katsuro Inoue"[6]	2017	"CCFinder—SW"	Kind -1, Kind -2	Token _ based	Implemented on different languages
"Seulbae Kim, Seunghoon Woo, Heejo Lee, Hakjoo Oh"[7]	2017	"VUDDY"	Kind-1, Kind-2	Text-based	C/C++
"Min Wang, Pengcheng Wang, Yun Xu"[8]	2017	"CCSharp"	Kind -4	PDG	C
"Chaiyong Ragkhitwetsagul, Jens Krinke"[10]	2017	NA	Kind -1, Kind -2, Kind -3	Textural	java
"Fang Lyu, Yaping Lin, Junfeng Yang"[11]	2017	NA	Kind -1, Kind -2, kind -3	Text-based	XML files
"Abdullah Sheneamer, Swarup Royc, Jugal Kalita" [9]	2018	NA	Kind -1, Kind -2, Kind -3, Kind -4,	Tree-based, Semantic (AST, PDG)	java
"Pengcheng Wang Jeffrey Svajlenko, Yanzhao Wu, Yun Xu, Chanchal K. Roy"[13]	2018	"CCAligner"	Kind -1, Kind -2, Kind -3,	Token_based	C language
"Chaiyong Ragkhitwetsagul, Jens Krinke Bruno Marnette"[14]	2018	"Vincent"	Kind -1, Kind -2, Kind -3	Matric-based(in image)	java
"Seulbae Kim, Heejo Lee"	2018		Kind -1, Kind -2	Text-based	C/C++
"Junaid Akram Zhendong Shi, Majid Mumtaz, Luo Ping"[15]	2018	"DroidCC"	Kind -1, Kind -2, Kind -3	Hybrid (token-based and text-based)	java
"Hamid Nasirloo, Fatemeh Azimzadeh"[16]	2018	NA	Kind -4	Hybrid (token-based and PDG)	C language
"Chunrong Fang, Zixi Liu, Yangyang Shi, Jeff Huang, Qingkai Shi"[17]	2020	NA	Kind-4	AST and CFG by embedding techniques	C++
"Yaroslav Golubev, Timofey	2021	NA	Kind-1, Kind-3		java language

Bryksin"[18]					
"Yaroslav Golubev, Viktor Poletansky, Nikita Povarov, Timofey Bryksin"[19]	2021	NA	Kind-3	Token_based	NA

5- Results and recommendations:

Figure 2 shows a chart that explains the kinds of clones that are detected by researchers. Most of the research detects kind1.Researchers detect the clone in the source code of different languages. Figure 3 summarizes the relationship between the research and the language of the source code. Most of the detected clone is in Java language, then C++ … etc.

It's noticed that the new languages that are used to write mobile applications till now are not included. So, it's a good field to do work in this era.

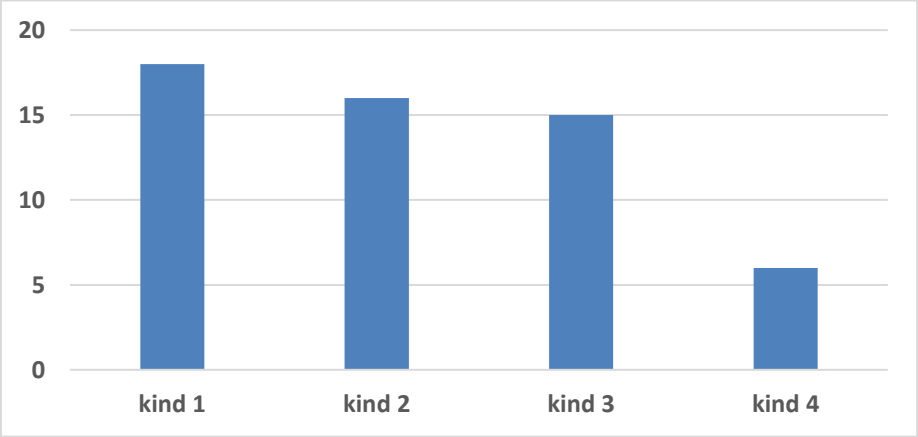

Figure 2- Relation between researches and clone kind.

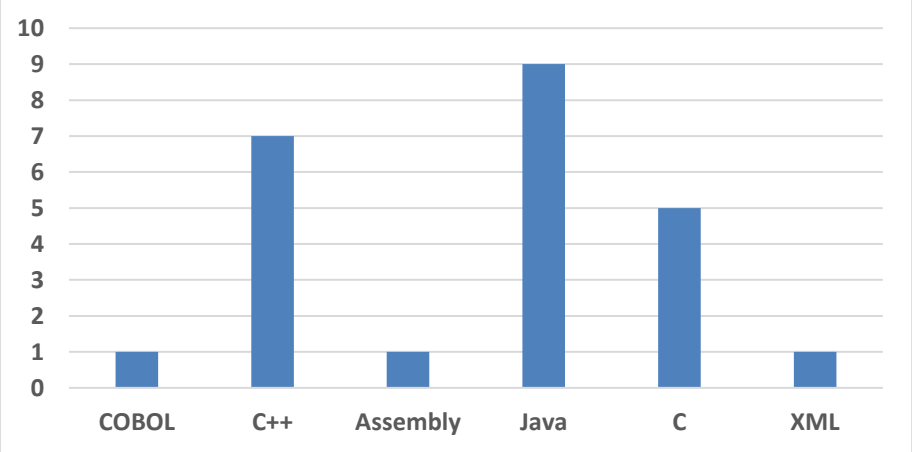

Figure 3-Relation between the language of the source code and the researches.

References

[1]- A. Kumar, R. Yadav and K. Kumar: A systematic review of semantic clone detection techniques in software systems.IOP Conference Series Materials Science and Engineering, January 2021,1022:012074 2021.

[2]- M. Sudhamani and L. Rangarajan: Code clone detection based on order and content of control statements, in Proc. 2nd Int. Conf. Contemp. Comput. Inform. (IC3I), 2016, pp. 59–64.

[3]- A. N. Runwal and A. D. Waghmare: Code clone detection based on logical similarity: A review, International Journal of Advanced Research in Computer and Communication Engineering, ISSN 2278-1021 ,(September 2017).

[4]- Roopam and G. Singh: To enhance the code clone detection algorithm by using hybrid approach for detection of code clones, International Conference on Intelligent Computing and Control Systems (ICICCS), 2017, pp. 192-198, doi: 10.1109/ICCONS.2017.8250708.

[5]- L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder: Cclearner: A deep learning-based clone detection approach,in Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME), Sep. 2017, pp. 249–260.

[6]- Y. Semura, N. Yoshida, E. Choi, and K. Inoue : CCFinderSW: Clone detection tool with flexible multilingual tokenization, in Proc. 24th Asia–Pacific Softw. Eng. Conf. (APSEC), 2017, pp. 654–659.

[7]- S. Kim, S. Woo, H. Lee, and H. Oh: VUDDY: A scalable approach for vulnerable code clone discovery, in Proc. IEEE Symp. Secur. Privacy (SP), May 2017, pp. 595–614.

[8]- M. Wang, P. Wang, and Y. Xu : CCSharp: An efficient three-phase code clone detector using modified PDGs, in Proc. 24th Asia–Pacific Softw. Eng. Conf. (APSEC), 2017, pp. 100–109.

[9]- A. Sheneamer, S. Roy, and J. Kalita: A detection framework for semantic code clones and obfuscated code, Expert Syst. Appl., vol. 97, pp. 405–420, May 2018.

[10]- C. Ragkhitwetsagul and J. Krinke: Using compilation/decompilation to enhance clone detection, in Proc. IEEE 11th Int. Workshop Softw. Clone (IWSC), vol. 11, Feb. 2017, pp. 8–14.

[11]- F. Lyu, Y. Lin, J. Yang, and J. Zhou: Suidroid: An efficient hardeningresilient approach to Android app clone detection, in Proc. IEEE Trustcom/BigDataSE/ISPA, Aug. 2016, pp. 511–518.

[12]- N.Saini, S.Singh ,and Suman: Code clones: Detection and management, Procedia Computer Science 132:718-727,(January 2018).

[13]- P. Wang, J. Svajlenko, Y. Wu, Y. Xu, and C. K. Roy: CCAligner: A token-based large-gap clone detector, in Proc. 40th Int. Conf. Softw. Eng., 2018, pp. 1066–1077.

[14]- C. Ragkhitwetsagul, J. Krinke, and B. Marnette: A picture is worth a thousand words: Code clone detection based on image similarity, in Proc. IEEE 12th Int. Workshop Softw. Clones (IWSC), Mar. 2018, pp. 44–50.

[15]- J. Akram, Z. Shi, M. Mumtaz, and P. Luo: DroidCC: A scalable clone detection approach for Android applications to detect similarity at a source code level, in Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC), Jul. 2018, pp. 100–105.

[16]- M. Sudhamani and L. Rangarajan: Code clone detection based on order and content of control statements, in Proc. 2nd Int. Conf. Contemp. Comput. Inform. (IC3I), 2016, pp. 59–64.

[17]- C. Fang, Z. Liu, Y. Shi, J. Huang, and Q. Shi: Functional code clone detection with syntax and semantics fusion learning, in ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, S. Khurshid and C. S. Pasareanu, Eds.NewYorkNYUnitedStates:ACM,2020, pp.516–527.

[18]- Y. Golubev and T Bryksin: On the Nature of Code Cloning in Open-Source Java Projects, in IEEE 15th International Workshop on Software Clones (IWSC),2021.

[19]- Y. Golubev, V. Poletansky, N. Povarov, and T. Bryksin: Multi-threshold token-based code clone detection,IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2021, pp. 496-500, doi: 10.1109/SANER50967.2021.00053.

[20] M. Kaur, S. Kaur, and B. Sohal: Review on Software Cloning and Clone Detection, International Journal of Control Theory and Applications 9(41):463-472,(December 2016).

[21]- T. Kamiya, S. Kusumoto, and K. Inoue: Ccfinder: a multilinguistic token-based code clone detection system for large scale source code, Software Engineering, IEEE Transactions on, vol.28, no.7, pp.654–670, (Jul2002).

[22]- E. Kodhai, S. Kanmani, A. Kamatchi, R. Radhika, and B. V. Saranya: Detection of type-1 and type-2 code clones using textual analysis and metrics, International Conference on Recent Trends in Information, Telecommunication and Computing, March 2010, pp. 241–243.

[23]- P. Batta and M. Himanshi: Hybrid technique for software code clone detection. INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY , ISSN: 2277–3061,(April 2012).

[24]- P.Charland, B. Fung and M.R. Farhadi: Clone Search for malicious code correlation. 2014, Retrieved from https://dmas.lab.mcgill.ca/fung/pub/CFF12ist.pdf .

[25]- A. Sheneamer and J. Kalita: Code clone detection using coarse and finegrained hybrid approaches, in Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS), Dec. 2015, pp. 472–480.

[26]- S. Chodarev, E. Pietriková, and J. Kollár: Haskell clone detection using pattern comparing algorithm, in Proc. 13th Int. Conf. Eng. Mod. Electr. Syst. (EMES), Jun. 2015, pp. 1–4.

[27]- Y. Yuki, Y. Higo, S. Kusumoto: A technique to detect multi-grained code clones. In Conference: IEEE 11th International Workshop on Software Clones (IWSC). 2017.