Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 3 Issue 1, January - 2016

Software Source Code Plagiarism Detection: A Survey

Mr. Bhramadeo Vishnu Deokate
ME-2nd Year Student Computer Engineering,
Vidya Pratishthan's college of engineering,
Baramati, Pune, India.
deokatebv@gmail.com

Prof. Dinesh Bhagwan Hanchate
Dept. of Computer Engineering,
Vidya Pratishthan's college of engineering,
Baramati, Pune, India.
dineshbhanchate@gmail.com

***Abstract*—Now a day's, big amount of data is avaliable on internet i.e information or source code. In this world of utilization, it is very difficult to find out the similarity or plagiarism, duplication of data in research, publications and practical program assignment in academics. Academics often use plagiarism detection tools to detect similar source-code duplication and similar files. In this paper, summary of the varius techniques and methods are explained how one should find out the plagiarisms in source code. A large organization or academic institute can easily find out the plagiarism in source code and research publications. The main problem is near-duplication of code that has been created by copying and modifying code with an editor that is code theft**.

Keywords—Software plagiarism detection, dynamic code identification.

I. INTRODUCTION

In the computer science field, there are number of probably that code theft may occurre. In the education field, students submit their projects work or the programming assignments, there may be possibility of duplication in source code. The manually plagiarism detection in the source code is a very difficult task. Mostly the people in computer science are using programming assignments of another one [7].

The plagiarism detection process consists of two parts. In the first part, it generates a representation from a given program. The intermediate representation is used for evaluating the similarity between two programs or projects. A token sequence is often used by intermediate representation. Plagiarism detection system uses the token sequence. In the second part, system evaluates similarity for each pair of programs [14].

Several techniques are developed for identifying similar code fragments in programs. These same fragments are referred as code clones. Some research has been dedicated to the methods for the detection of similar code fragments in programs.

Software projects with similar codes, which may be introduced by many commonly adopted software development practices, and due to reusing a generic framework, following a specific programming pattern, and directly copying and pasting code. Some times, these practices can decrese the productivity of software application [15].

Source code plagiarism is easy to do, but it is not easy to detect. Usually, when students are solving the same problem by using the same programming language source code, there is a high possibility that their assignment solutions will be more similar. Strategies of source code modification exist that can be used to mask source code plagiarism. Examples of such strategies are renaming identifiers and combining several segments copied from different source code files. These modifications increase the difficulty in recognizing plagiarism [9].

Plagiarism in coding is not entirely a new phenomenon. The issue has been discussed and studied previously by researchers to identify the severity of the problem and amount of factors contribute to the act of plagiarism. In programming assignment, plagiarism does necessarily only involve copying the source code but input data and interface designs that can also be considered as possibility of plagiarizing content [1].

II. LITERATURE SURVEY

Sr. No.	Author and Year	Outcome	Methods / Algorithm	Tools	Purpose
1.	Brenda S. Baker 1995	Work on source code is exactly matched or near matched.	Text matching, parameter matches		Much duplication has been found.
2.	T. Kamiya 2002	Code clone detection technique.	Token based code clone detection method	CCFinde	Detecting the similar term into source code.
3.	Mark Gabel 2004	Identifying similar code fragments in programs.	Clone Detection Method	CloneDR	Detecting fragments into single program.
4.	C.Collberg 2007	Identify duplication in java code.	Java code obfuscetor		To cheaking the code duplication.
5.	Enrique Flores, 2011	To detect cross-language reuse between source codes.	Character N-Grams comparison model		To detect cross-language reuse between source codes.
6.	Ameera Jadalla & Ashraf Elnagar 2007	To create a one engine that detect the source code plagiarism in Java.	Clustering, N-Gram, Tokenization	JPlag	Performance of the system, pair wise similarity measurement.
7.	A. Bugarín, M. Carreira	This paper shows how software tools detect the plagiarism.		JPlag and Turnitin	There are mainly two tools used, One is JPlag and another is Turnitin.
8.	Dong-Kyu Chae	Main aims of this research is to create a software plagiarism detection system.	API- labeled control flow graph		Accuracy and credibility in a reasonable computation time.

III. RELETED WORK

The goal of plagiarism detection approaches is showing potentially plagiarised source code pairs. A system determines which case pairs are likely to be plagiarised by analysing the similarity levels between texts in the programs code. If the similarity level between a case pair is high, the system indicates the case pair is suspicious and suggests to the user that this pair may require further investigation [8].

Tools	JPLAG	SIM	MOSS	PLAGGIE
Open source	NO	YES	NO	YES
Local/Online	Web	Local	Web	Local
Codebase /File	Code base	File base	Code base	Code base
Language Support	6	5	23	1
Founded Year	1996	1989	1994	2002
Founded By	Guido Malpohl	Dick Grune	Aikenetal	Ahtiaine netal

Table: comparison of four plagiarism detection tool

Detecting Source Code Re-Use: It detect source code reuse in the many programming languages or projects. The main target of this is to provide new technologies to detecting source code duplication. Using these tools, it decides whether the source code has been reused or not. DeSoCoRe compares two source codes at the level of functions and method even when written in different programming languages [6].

Plagiarism Detection Engine For Java Source Code: Here in this research article, authors Ameera Jadalla & Ashraf Elnagar developed PDE4Java model that is basically used for plagiarism detection in java source code. Method used in this research is data mining, clustering, N-Gram, tokenization. At the end of the research it shows performance of the "system pair wise similarity" in programing language [15].
Detecting source code reuse across programming languages is based on character. The problem of cross-programming language reuse of source code at document and fragment levels in first part. In the second parts, fragments of source codes are compared with detecting only those fragments in the source code.

Plagiarism Detection Tools:

Software Similarity Tester: SIM is used to detect plagiarism in source code written in Java, C, C++, Pascal, .NET and python. This tool is also used to check similarity between source codes. SIM converts the source code into strings of token and then compare these strings by using dynamic programming string alignment method. This method is used in string matching. The alignment is very expensive and exhaustive computationally for all applications because for large source code repositories SIM is not scalable. The source code of SIM is available publically but it is no more actively supported.

Measure of Software Similarity: MOSS tool is available free to use in academics and it is accessible as an online service. Moss support Ada programs, Java, C, C++, plain text and Pascal. The MOSS tool is

also support to UNIX and windows operating systems. It uses a string matching algorithm to divide the source-code programs into k-grams, hash them, select a subset of these hashes as fingerprints and finally compare these fingerprints.

Plague: The earliest structure-oriented system is Plague. Plague only support programs written in C. This tool works in numbers of steps. The first step source code is converted into structure files. After this Plague use Heckel algorithm to compare generated structure files of first step. The algorithm is basically designed for plain text files and it is introduced by Paul Heckel. Plague's detection results are returned in the form of lists .Plague returns use an interpreter to process this list to show results in a way, so that common user can understand easily.

Yet Another Plague: YAP was developed based on Plague with some enhancements. The first version was created by Michael Wise. Then it was optimized into YAP2. YAP2 came into market a bit later after YAP1 and finally the YAP3 of YAP family was developed in 1996. The final version YAP3, which can also be used to detect text plagiarism. YAP shows result in a plan text file. If token pairs have percent match value larger then lowest value set by user then the matching pair will be consider as plagiarized pair.

.

Data Sources:

1) Java project: In java project function, class, methods are repeated so, that project accuracy decreses.
2) Java Assignment: Number of times java assignment are repeated in a education acedima.
3) Java Tutorial: The practise program or a sample codes should be duplicated.

Applications:
1) Education System: Plagiarism is a big problem in education system. Academics often use plagiarism detection tools to detect similar source-code files or java program. Once similar files are detected in a project the academic precedure its duplication or a reuse. The investigation process which involves identifying the similar source-code fragments proving plagiarism. So the plagiarism process is necessary in that system.

2) **Corporate**: Code duplication has been practiceing from the earliest days of programming. Developer has always reused part of code, templates, functions, and procedures. Code reuse possibility is a recognized in industry. Developer works on project on time code reuse in program so, to degrade the project quality.

3) **Publication system**: Redundant publication in the paper some part is similar to a published paper by the same author. Publishing without acknowledging the source and without obtaining the permission of the original copy right holder comes under this categery. There may be differences between the original and the second paper such as a new title or a modified abstract. It violates generaly copyright as in most of the time, it creates problem.

IV. CONCLUSION

A survey on plagiarism detection system in a source code and project has been introduced. The information of plagiarism problem is studied to education, publication, corporate and social websites. The need of plagiarism detection system is must now a days to improve academic integrity, and also uniqueness in a project for increasing the quality of programs and educational assignment. So that the similar source code fragments should not occur for proving the plagiarism.

V. ACKNOWLEDGMENTS

This paper would not have been written without the valuable advices and encouragement of Asst. Prof. D.B. Hanchate, guide of ME Dissertation work. We thank to Prof. P. M. Patil and Prof. S. S. Nandgaonkar, Head of Department and Hon'ble principal Prof. V. U. Deshmukh, for their valuable support and for giving an opportunity to work on Software Source Code Plagairism Detection.

VI. REFERENCES

[1] B. S. Baker, "On finding duplication and near-duplication in large software systems," in Proceedings of 2nd Working Conference on Reverse Engineering (WCRE '95), 1995, pp. 86–95.
[2] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees." in Int. Conf. on Software Maintenance, 1998.
[3] K. Kontogiannis, M. Galler, and R. DeMori, "Detecting code similarity using patterns." in Working Notes of 3rd Workshop on AI and Software Engineering, 1995.
[4] J. Krinke, "Identifying similar code with program dependence graphs." in Proceedings of Eighth Working Conference on Reverse Engineering (WCRE '01), 2001, pp. 301–309.
[5] T. Kamiya, S. Kusumoto, and K. Inoue., "CCFinder: a multilin-guistic token-based code clone detection system for large scale source code." IEEE Transactions on Software Engineering, vol. 28,no. 7, pp. 654–670, 2002.
[6] M. Gabel, L. Jiang, and Z. Su, "Scalable detection of semantic clones," in Proceedings of the 30th International Conference on Software Engineering (ICSE'08), 2008, pp. 321–330.
[7] L. Jiang, Z. Su, and E. Chiu, "Context-based detection of clone related bugs," in Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT symposium on the Foundations of Software Engineering, ser. ESECFSE '07, 2007, pp. 55–64.
[8] L. Jiang, G. Misherghi, Z. Su, and S. Glondu, "DECKARD: Scalable and accurate tree-based

detection of code clones," in Proceedings of the 29th International Conference on Software Engineering (ICSE '07), 2007, pp. 96–105.
[9] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," The Univeristy of Auckland, Tech. Rep.148, Jul. 1997.
[10] C. S. Collberg, C. Thomborson, and D. Low, "Manufacturing cheap, resilient, and stealthy opaque constructs," in Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '98), 1998, pp. 184–196.
[11] C. Collberg, G. Myles, and A. Huntwork, "Sandmark–a tool for software protection research," IEEE Security and Privacy, vol. 1, no. 4, pp. 40–49, 2003.
[12] M. Madou, L. Van Put, and K. De Bosschere, "Loco: An interactive code (de)obfuscation tool," in Proceedings of the 2006 ACM SIG-PLAN symposium on Partial evaluation and semantics-based program manipulation (PEPM '06), 2006, pp. 140–144.
[13] H. Tamada, K. Okamoto, M. Nakamura, and A. Monden, "Dynamic software birthmarks to detect the theft of Windows applications," in Int'l Symp. On Future Software Technology (ISFST), October 2004.
[14] F. Zhang, Y. Jhi, D. Wu, P. Liu, and S. Zhu, "A first step towards algorithm plagiarism detection," in Proceedings of the 2012 International Symposium on Software Testing and Analysis (ISSTA '12).ACM, 2012, pp. 111–121.
[15] L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu, "Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection," in Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014), November 2014.