Similarities in Source Code

Marek ROŠTÁR*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
rostarmarek@gmail.com

In current day and age with the development of software there is an increase of problems concerning plagiarism, but also it is quite common to see repeated use of parts of source code. These two issues are the main reasons to look into source code comparison, since it can detect most of plagiarism attempts and also highlight which parts of our source code are we using repeatedly. Such parts we may consider o turn into some sort of library or plugin to improve our source code quality.

Plagiarism is these days one of the main problems of the academia, affecting not only text documents but also most of intellectual property including source codes. It is not uncommon that students inspire themselves with some work they found on the Internet.

In this work we focus on detecting plagiarism in source code specifically (considering its special features in comparison to the standard text documents). Methods used to detect plagiarism in source code differ from methods used to detect plagiarism in text documents [1, 2], since the text in source code does not carry only meaning but some sort of function as well.

Plagiarists try to deceive anyone viewing their work with a multitude of different plagiarism attacks, which for plagiarism in source codes, the basic ones are as follows: altering comments in source code, altering whitespaces present in source code, altering names of variables, altering the order of parts of the source code, altering algebraic expressions in source code, translating source code into another programming language and extracting parts of source code.

We detect these attacks using different methods that we divide into four groups: text based methods, token based methods, tree based methods and semantic based methods. These methods vary in their vulnerability to different attacks, with text based methods being most vulnerable and semantic based methods being least vulnerable to the attacks mentioned before.

In this work we focus on tree based methods, since we are using them in our proposed solution. Tree based methods are based on converting source code into a data

* Supervisor: Michal Kompan, Institute of Informatics, Information Systems and Software Engineering

Spring 2016 PeWe Workshop, April 2, 2016, pp. 25–26.

structure of tree and after the conversion comparing the trees instead of source codes themselves. This is achieved by doing a lexical analysis on the source code to tokenize it and after the lexical analysis comes syntactical analysis that takes blocks of source code and transforms them into nodes. The nodes are hashed and then compared by node to node basis.

Advantages of this method are that it is resistant to plagiarism attacks and it can find behavioral changes, which can be used for finding malicious source code [3]. Its disadvantages are time consumption and addition of new data structures.

When we are working with source code we can remove for us unimportant parts and thus create an abstract source code. We use this sort of preprocessing often when we compare source code to find similarities in given source codes. There are multiple levels of abstraction, and we can determine how strong abstraction we need based on how big is the source code [4]. Thanks to abstraction we can vastly reduce the volume of the source code and with that speed up the process of comparing the source code.

Our proposed solution is to compare source codes using abstract syntax tree and evaluate if abstracting source code before comparing it with this method is viable. There will be three levels, ranging from removing only comments and unifying white spaces to the highest level, in which we will replace strings values, remove variable declarations, unify the names of variables and greatly reduce the volume of the source code.

Amended version was published in Proc. of the 12th Student Research Conference in Informatics and Information Technologies (IIT.SRC 2016), STU Bratislava.

References

[1] Alzahrani, S. M., Salim, N., & Abraham, A.: Understanding plagiarism linguistic patterns, textual features, and detection methods. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, (2012), vol. 42, no. 2, pp. 133-149.

[2] Beth, B.: A Comparison of Similarity Techniques for Detecting Source Code Plagiarism. (2014).

[3] Neamtiu, I., Foster, J. S., & Hicks, M.: Understanding source code evolution using abstract syntax tree matching. *ACM SIGSOFT Software Engineering Notes*, (2005), vol. 30, no.4, 1.

[4] Park, S., Ko, S., Choi, J. J., Han, H., & Cho, S.-J.: Detecting Source Code Similarity Using Code Abstraction Categories and Subject Descriptors. *Proc. of the 7th Int. Conf. on Ubiquitous Information Management and Communication - ICUIMC '13*, (2013), 1-9.