Indian Journal of Science and Technology, *Vol 9(39), DOI: 10.17485/ijst/2016/v9i39/96195, October 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Falsification: An Advanced Tool for Detection of Duplex Code

U. Janardhan Reddy[1], Kuncham Siva Krishna Rao[2], M. Nirupama Bhat[1] and K. B. S. Sastry[3]*

[1]Department of Information Technology, VFSTR University, Vadlamudi, Guntur - 522213, Andhra Pradesh, India; ummadi.janardhan@gmail.com, nirupamakonda@gmail.com
[2]Department of Computer Science and Engineering, KL University, Vaddeswaram - 522502, Andhra Pradesh, India; shivasob12@kluniversity.in,
[3]Department of Computer Science, Andhra Loyola College, Vijayawada - 520008, Andhra Pradesh, India; sastrykbs@gmail.com

Abstract

Objective: Today the most important objective in the programming is cloning the code. It is becoming the most dangerous issue in the software programming. **Method**: the proposed method falsification is the tool which identifies the similar code or duplicate code or duplex code from various sources. It detects the duplicate code from languages such as C++, C#, Java etc. In the existing system, there are no of tools to detect the duplex codes with minimum functionality. **Application:** Falsification is integrated with astute compiler developed for oops languages and advanced ontology editor. In this paper, an algorithm called false code detection is implemented for better results to detect the duplicate code from various sources. **Findings:** Results will show performance of the falsification and working of astute compiler and advanced ontology editor.

Keywords: Astute, Duplex, Falsification

1. Introduction

Plagiarism is most widely used by the students of many universities. Code plagiarism is also called as code cloning[1]. Copying the code is a crime. In other words, cloning the code. It is very harmful to the software quality and maintenance of the software is difficult. Detecting the plagiarism is most widely done by the many researchers[2]. There are some levels of code cloning occurs in the software development. If the cloning of code is high the software may have lost the quality. Changing the synonyms copying the content from many sources and modifying the text is under plagisrism[3]. During the software development

copy and paste the code occurs very commonly. The result shows the similar code in existing software development.

The fact in software maintenance is the most expensive one in the entire project development system. According to the report of researchers the Multi National software companies spend a huge amount for maintaining the existing systems[4]. Maintenance or services means after completion of project according to the client requirement every time changes occur and if any problems in the software support should be provided by the companies for better services[5].

Code clones becomes more important topic among software maintenance researchers[6]. Various researchers contemplate clones to be unsafe[6–12] due to the conviction that conflicting changes increment each repairs effort and therefore the likelihood of presenting defects. Yet, completely different researchers do not discover experimental confirmation of harm, or perhaps came upon biological research as a big programming building strategy to beat idiom impediments or to follow regular elements of the code. It's not nevertheless clear that of those two dreams wins, or whether or not the proper idea depends on the software system which is similar.

*Author for correspondence

2. Related Work

Juergens[13] investigate, from the five software development companies the manual checking of clones in a source system needs to be changed the clones that are very frequent which lead to huge number of exceptions in software development.

Johnson[14] studied history of clones (between two versions of the GCC compiler). From the past few years many research has been done in larger systems such as large telecommunication system and the Linux kernel. Geiger[15] expected that many number of clones are shared between records; together more documents are changed.

Hotta[16] showed a different technique on the impact of clones in software maintenance activities to calculate the changed frequencies of the duplicated and non-duplicated code fragments. According to the research, the clones that are identified do not create any issues at the time of software maintenance phase.

In[17] proposed a model of clone genealogy on clone evolution. According to the research, extracting of clones may not always improve software quality based on the revisions of two medium sized Java systems during their research.

In[18] reported Type (I) code clones that can be varied consistently during services measured by Simian[19] (a code clone detector) and diff (a file comparison utility) on Java, C and C++. He also found that half of lifetime of clone groups consistently changing.

3. Research Questions

In this section, some of the research questions have been addressed to find out the exact solution for the detection of cloning code and to know how the cloning detection have been done. According to the research detection of cloning code or duplicate code in programming languages few questions have been proposed.

1. How the detection starts from programs?
2. What is the process used to detect cloning code?
3. Is the detection done like compiler?

3.1 Existing Cloning Tools

In this section, the number of cloning tools have been introduced.

PMD (www.PMD.sourceforge.net/) is developed in Java source code which identifies the difficult problems like copied code, identified bugs, and final code. This will provide some measurements for limitations of copied code from the source code.

Bauhaus (www.bauhaus-stuttgart.de/) gives backing to break down and recuperate a system's software architecture; a few support undertakings are bolstered as the induction of diverse perspectives on the building design of legacy frameworks, ID of reusable parts, and estimation of progress effect. The Bauhaus module for discovering copied code searches for three sorts of clones: segments of indistinguishable code, their variety with distinctive variable names and identifiers, and parts of indistinguishable code with included or evacuated articulations.

GoogleCodeProAnalytix is a Java testing device for Eclipse designers who are worried about enhancing programming quality. The primary components are identified with code investigation, measurements calculation, JUnit test era, reliance examination, and comparable code investigation. For clone discovery, the instrument offers three sorts of inquiry: (1) code that can be refectory, (2) code that contains conceivable renaming blunders, and (3) fair appears to be comparable. We picked the last alternative to discover however many copied code events as would be prudent.

4. Falsification

It is the tool, used to detect the cloning code which is integrated with plug-in. In this paper, the proposed tool integrated with astute compiler developed for oops languages and advanced ontology editor. Programming languages like C, C++, Cobol, C# and all the object oriented programming languages support.

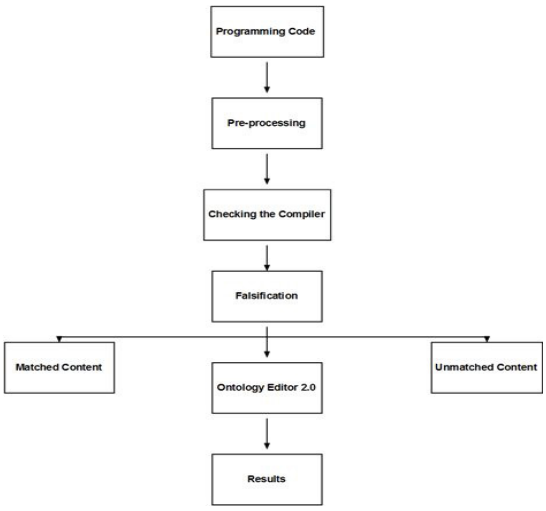

Figure 1. Falsification architecture.

To compile the code, the proposed system integrated with astute compiler. This is the compiler works as the common compiler which will check the syntax of every supporting language of falsification. In real world, for every language there should be a individual compiler for every programming language. But in this tool a common compiler compiles the given code in editor. Common compiler is integrated with falsification or it is the part of falsification. In this paper, an algorithm called false code detection algorithm is implemented for better results. The FCDA process three stages.

4.1 Code Processing

In this stage, the code is ready for the cloning detection. Using ontology editor, the compiler checks every line of code with all given parameters, methods, variables etc. Now, the code is divided into three parts, a.) Checking all the variables, b.) Checking all the methods, functions, classes etc, c.) Checking all the other statements.

4.2 Syntax Processing

Before showing the output, firstly we check the syntax of code given in the ontology editor. This will finalize the given code belongs to which language. If any error occurs, the error will be shown with the red line like code interpreter. After that, the code is ready without any errors and the detection of cloning code is started.

4.3 Searching Process

In this stage, the compilation starts in the ontology editor and as well as search the false detection of given code

User Code	Cloning code detection

#include <iostream.h>
#include <conio.h>
 	
void main()
{
clrscr();
int result;
float cdegree,fdegree;
cout << "1.Celsius to Fahrenheit" << endl;
cout << "2.Fahrenheit to Celsius" << endl;
cout << "Choose between 1 & 2 : " << endl;
cin>> result;
if (result == 1)
{
cout << "Enter the temperature in Celsius : " << endl;
cin>> cdegree;
fdegree = (1.8 * cdegree)+32;
cout << "Temperature in Fahrenheit = " << fdegree <<
endl;
}
else
{
cout << "Enter the temperature in Fahrenheit : " << endl;
cin>> fdegree;
ctemp=(fdegree -32)/1.8;
cout << "Temperature in Celsius = " << cdegree <<
endl;
}
getch();
}
	#include <iostream.h>
	#include <conio.h>
	
	void main()
	{
	clrscr();
	int choice;
	float ctemp,ftemp;
	cout << "1.Celsius to Fahrenheit" << endl;
	cout << "2.Fahrenheit to Celsius" << endl;
	cout << "Choose between 1 & 2 : " << endl;
	cin>>choice;
	if (choice==1)
	{
	cout << "Enter the temperature in Celsius : " << endl;
	cin>>ctemp;
	ftemp=(1.8*ctemp)+32;
	cout << "Temperature in Fahrenheit = " << ftemp <<
	endl;
	}
	else
	{
	cout << "Enter the temperature in Fahrenheit : " << endl;
	cin>>ftemp;
	ctemp=(ftemp-32)/1.8;
	cout << "Temperature in Celsius = " << ctemp << endl;
	}
	getch();
	}

Figure 2. Comparison of two source files.

from various sources. Here the comparison is on variables, classes, methods etc. After showing the stage wise results, the mean for all the 3 stage results for the final output is calculated.

5. Experimental Results

In this project, two sample code editors are present which consists of two source code files developed in JAVA programming language and Netbeans IDE. From these files we have some variations and similarities in the code as shown in Figure 1. In the starting, the white spaces and comments are removed from two files.

In the C++ code, the cloning code detected every line of copied code from various sources and highlighted the copied code in yellow lines (Figure 2). Falsification finds the accurate results of cloning code with respect to functions, methods, parameters etc. It is identified that our tool checks each and every line of code given in the editor.

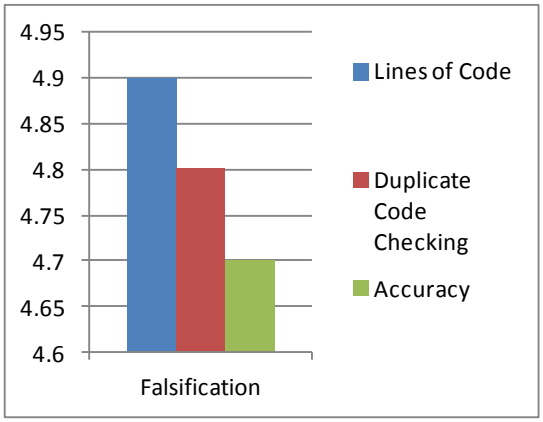

Figure 3. Performance of falsification.

5.1 Features of Falsification

1. No trees, graphs are not included.
2. Only line by line checking is done in falsification.
3. Detects huge lines of code.
4. Support structured code oops code.
5. Compares each and every line from various sources.
6. Finding similar code patterns.
7. Checking with common compiler.
8. Ignores whitespaces and comments.
9. Find the changed cloned methods, variables etc.
10. Checking thousands of lines of code at a time.

6. Conclusion

In this paper, an integrated astute compiler is developed for oops languages and advanced ontology editor. An algorithm called False Code Detection Algorithm (FCDA) is implemented for better results. Falsification supports the c, c++, c# and java coding languages for detection of cloning codes. The performance is very high. Thousands of lines of code can be detected with in short span of time with the falsification. In future, integrated proxy compiler has to be introduced to increase the duplex code detection with very little span of time.

7. References

1. Karuna P, Preeti M. Global plagiarism management through intelligence of hawk eye. Indian Journal of Science and Technology. 2016 Apr; 9(15). DOI: 10.17485/ijst/2016/v9i15/92113.
2. Al-Shamery ES, Gheni HQ. Plagiarism detection using semantic analysis. Indian Journal of Science and Technology. 2016 Jan; 9(1). DOI: 10.17485/ijst/2016/v9i1/84235.
3. Garg U, Goyal V. Maulik: A plagiarism detection tool for Hindi documents. Indian Journal of Science and Technology. 2016 Mar; 9(12). DOI: 10.17485/ijst/2016/v9i12/86631.
4. Higo Y, Ueda Y, Kamiya T, Kusumoto S, Inoue K. On software maintenance process improvement based on code clone analysis (SMPIBCC). Product Focused Software Process Improvement. 2002 Dec 18; 2559:185–97.
5. Pigoski TM. Encyclopedia of Software Engineering, Maintenance. John Wiley and Sons; 1994.
6. Bettenburg N, Shang W, Ibrahim WM, Adams B, Zou Y, Hassan AE. An empirical study on inconsistent changes to code clones at the release level. Science of Computer Programming. 2012 Jun; 77(6):760–76.
7. Baker BS. On finding duplication and near-duplication in large software systems, Proceedings of the 2nd Working Conference on Reverse Engineering, WCRE'95, IEEE Computer Society; 1995.
8. Baxter ID, Yahin A, de Moura LM, Sant'Anna M, Bier L. Clone detection using abstract syntax trees. ICSM'98: Proceedings of the 14th IEEE International Conference on Software Maintenance, IEEE Computer Society; 1998.
9. Geiger R, Fluri B, Gall HC, Pinzger M. Relation of code clones and change couplings, FASE'06: Proceedings of the 9th International Conference of Funtamental Approaches to Software Engineering, Springer; 2006.

10. Do code clones matter? ICSE'09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, IEEE Computer Society; 2009.
11. Kamiya T, Kusumoto S, Inoue K. Ccfinder: A multilinguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering. 2002; 28(7).
12. Kontogiannis K, de Mori R, Merlo E, Galler M, Bernstein M. Pattern matching for clone and concept detection. Automated Software Engineering. 1996; 3(1–2).
13. Lozano A, Wermelinger M. Assessing the effect of clones on changeability. ICSM'08: Proceedings of the 24th IEEE International Conference on Software Maintenance, IEEE; 2008.
14. Hotta K, Sano Y, Higo Y, Kusumoto S. Is duplicate code more frequently modified than non-duplicate code in software evolution? An Empirical Study on Open Source Software. Proceedings EVOL/IWPSE; 2010.
15. Kim M, Sazawal V, Notkin D, Murphy GC. An empirical study of code clone genealogies. Proceedings ESEC-FSE; 2005.
16. Krinke J. A study of consistent and inconsistent changes to code clones. Proceedings of WCRE; 2007.
17. Simian similarity analyser [Internet]. Available from: http://www.redhillconsulting.com.au/products/simian.
18. Veeranjaneyulu R, Srinivasul S. Dupian: Duplicate code analyzer using ontology. International Journal for Development of Computer Science and Technology. 2015 Oct–Nov; 3(7).