International Journal of Security and Its Applications
Vol.7, No.5 (2013), pp.269-278
http://dx.doi.org/10.14257/ijsia.2013.7.5.25

Preventing and Detecting Plagiarism in Programming Course

Wang Chunhui, Liu Zhiguo and Liu Dongsheng

Computer & Information Engineering College, Inner Mongolia Normal University
Huhhot, China
ciecwch@imnu.edu.cn

Abstract

Student plagiarism is epidemic in universities. In computer programming education process, it is very common that students copy or modify other's code as their own work. Because every course having numerous assignments, detected plagiarism will be very difficult and extremely time consuming. How to prevent the plagiarism in programming course is an important problem in education. This paper analyzes and expatiates the reasons and the methods about the code's plagiarism, and thinks there are two phases in preventing this plagiarism: one is preventing plagiarism from occur, the other is to detect cases of plagiarism when the preventative measures fail. Preventing plagiarism methods mainly include the valid course assignment design and to forbid the electronic copy. This paper describes a code's editor software which has been implemented use Java. When the preventative measures fail, this paper describes an automatic tool to help instructor find the suspicious targets. These phases' aim is to cut down the plagiarism and improve the ability of the student' programming

Keywords: *code plagiarism detection, anti-plagiarism editor, software of code plagiarism detection*

1. Introduction

Plagiarism is epidemic in programming courses [1-2]. The plagiarism behavior usually appears in coping other students' programming assignments or to obtain more assists from Internet or other electronic library resource [2]. The programming courses need numerous assignments to get better educational aims. These assignments usually are program codes. The electronic nature of these assignments means copying others' work is very easy. If students copy others' assignments, it is difficult for them to learn programming, so preventing and detecting plagiarism in programming courses is a serious issue [3].

This unethical student behavior requires instructors to address the plagiarism that occurs in their classes. Instructors need evidence to prove their students plagiarize others. Then they can penalize them and then prevent plagiarism. If the instructor manually compare every two different students' code to find the plagiarism parts, this procedure is tedious and expensive to the time cost. It also is difficult when the number of students' assignments is considerable.

Currently many automatic plagiarism detection systems have been developed [4-6]. These systems can find some similar code quickly to help instructor find plagiarism. However, because the tyro assignment's code usually is short, simple and regular to realize method, then the detection tools will report a lot of similar code. That is some similarity maybe not plagiarism but due to the simple assignment. In introduction programming course, the simple assignment is general. So if instructors hope to find the evidences in every pairs of plagiarism code, they must attentively analyze the tool's reporting results to assess these similar code

ISSN: 1738-9976 IJSIA

assignments and find the plagiarism behavior. If the number of students' assignments is large, the workload is still considerable. So the automatic detection results usually are of less reference value due to the above reasons for introduction programming course [7]. In addition automated tools can only detect plagiarism when it does occur, so we cannot educate students based solely on the results of automatic tools detection.

In this paper, plagiarism and methods for dealing with plagiarism are discussed. This paper also concentrates on the ways of how to prevent the plagiarism in the introductory programming course. We can use the automated tools to help the instructors. But it is not enough, we needs other ways to prevent the students' plagiarism, it is more important than detection after plagiarism.

2. Background

About the plagiarism, some of the most commonly reported causes are [8, 14, 15]:

- a lack of time to undertake the assessment task;
- fear of failure and error;
- assessment tasks which are too difficult and can not finish;
- inadequate resources (hardware, software, library, etc.);
- a lack of educational merit of the assessment task.

The temptation to plagiarize in the academic setting is no limited to students. So we must detect the plagiarism. Parker and Hamblen [9] has defined that a plagiarized program produced from another program with a small number of text edit operations but no detailed understanding of the program required. In the student programs, the text edit operations include:

- Changing comments or formatting
- Changing identifiers
- Changing the order of code
- Replacing expressions with equivalents
- Adding redundant statements or variables
- Replacing procedure calls by the procedure body

In fact, common plagiarism in student programs is to copy the whole code's files or copy the main sentence from other electronic resource, because this method makes the copying process more labour-intensive. Figure 1 shows the proportion of the different plagiarism methods in student program. To different plagiarism methods, we can see that the proportion of simple copy accounts for great majority.

There are various systems to detect plagiarism in computing course-work. Based on which characteristic properties they employ in their comparisons, these systems can be roughly grouped into two categories: attribute-counting systems and structure-metric systems.

The attribute-counting system [10-13] usually counts the number of unique operators and operands, and the number of loops and procedures in the code; then calculate the value of these attributes vector. And the value can show the similarity level of two programming. These methods need not analyze the code's structure, so the system can fast get the plagiarism detected result. However, these methods can not find some plagiarism behaviors, such as

replacing procedure calls by the procedure body, or replacing some codes which the function is same but sentence is different.

The Structure-metric system usually extracts and compares representations of the program structure. therefore it gives an improved measure of similarity and is more effective technique to detect program plagiarism [4]. There are some servers on the web which detect plagiarism. For example, JPlag [5] at Karlsruhe University finds pairs of similar programs, and the MOSS server at Berkeley [6] looks for similar code sequences in a set of programs; each system creates a web page where the instructor can see which ones are suspiciously similar.

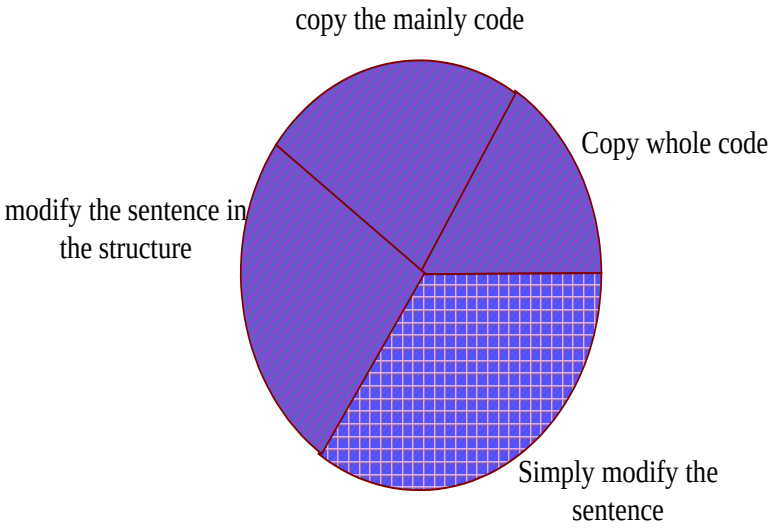

Figure1. The Proportion of the Different Plagiarism Methods in Student Program

3. Plagiarism Prevention and Detection

The approaches to minimize the incidence and impact of plagiarism fall into two complementary categories: preventing plagiarism from occurring, and detecting cases of plagiarism when the preventative measures fail.

We use an anti-plagiarism system to prevent the most common electronic copy, and then use the Jplag online server to detect the plagiarism after the students submit the assignments.

The anti-plagiarism has been developed by us using Java. It works in two phases:

In the first phase, a student needs log in the system using his/her own information (such as student ID or name), then the student can look at the assignment caption and then edit the code. The system will monitor the student's actions during the development of their submitted assignment. In this development phase, we offer an editor, and the editor can prevent the electronic copy. The major methods include two sides: one is to forbid the paste coming from other editors. The other is to add encryption based on individual information in the code, and the aim is to prevent other's work from opening.

In the second phase, the assignment code has been submitted, we use Jplag to detect the plagiarism and collect copied materials.

Figure2 shows the architecture of the anti-plagiarism.

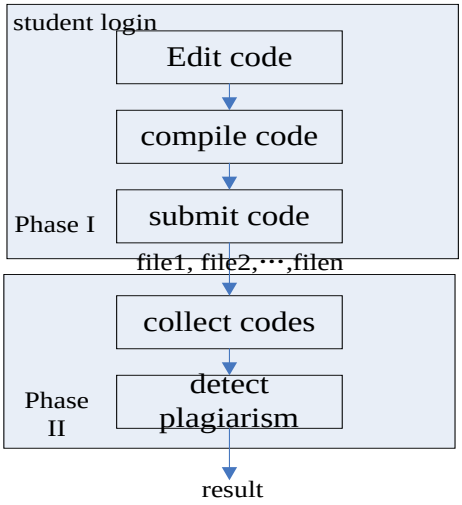

Figure 2. The Architecture of the Anti-plagiarism

3.1. Preventing Plagiarism from Occurring

Preventing plagiarism from occurring is an effective method in programming course education. The instructors prevent the code plagiarism by means of the coursework design, virtue education and software assist. The students must finish the assignments by themselves. In addition, it also can reduce the time cost to detect plagiarism.

We can prevent plagiarism according to the following methods:

- Honest and ethical education: we need educate students about the nature of plagiarism, and the reasons why it is unacceptable. We must let students know that copy other's work will fail to all programming course. Because these course are relational with each other.
- Warnings and penalties: Before teaching, we warn the students they should be penalized if they be found plagiarism. The penalties usually include getting zero, failing in this course, having more assignments and so on.
- Coursework design: Assignment should be changed from year-to-year to avoid the re-use of solutions from previous years, and should contain unique elements to impede the inclusion of generic code from online sources. In addition to, assignment should be not too difficult and be interest, thus the students should be willing to do by themselves.
- Monitor the student's actions during the development of their submitted work: we use the automatic tool to help tutors monitor the student's actions during achieve their submitted work. It is an anti-plagiarism, show in the figure4. The tool can prevent the copy coming from other editor tools. It should prevent the electronic copy. In addition, we will force the students must use the tool to submit the work because it will add the individual information when they submit the work. The system can prevent the student submit the other's code, because every individual information is different and the editor add encryption based on individual information in the code, so it can find these plagiarism after one submit other's work.
- Audits the work after its submission: The electronic copy is most commonly plagiarism in programming course. We found that some students submit others code

and did not modify. Using the anti-plagiarism editor can add the individual information in the assignment's code, and the editor can automatic find these code that the individual information is different with the submission, and then we can confirm that he/she summit other's work and he/she is plagiarism. This work will be finished after the students submit their assignments.

3.2. Detection Plagiarism when the Preventative Measures Fail

We can not ensure that student have no plagiarism during preventing stage. So we must detect plagiarism when the preventative measures fail. We can use the automatic tool to detect the plagiarism. Jplag is a system that finds similarities among multiple sets of source code files. Current version of Jplag supports Java,C,C++. We use the system to help detect plagiarism of student's assignments.

The plagiarism detection works four phases [16]:

In the first phase, the instructor collects every student's concrete assignment into one folder.

In the second, the instructor uses the Jplag automatic detection tool to detect plagiarism, and gets the results about every pair code's similarities.

In the third, the instructor looks over the report coming from the Jplag and confirms the suspicions.

In the last, the instructor investigates the suspicions to test and verify the plagiarism.

Figure 3 is shown the plagiarism detection procedure.

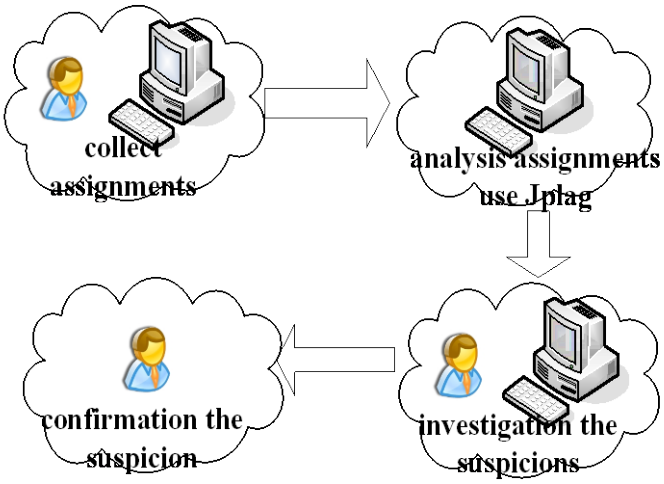

Figure 3. Plagiarism Detection Procedure

3.2.1. Collection the Student's Assignments: We use the anti-plagiarism editor system in this step. The system's main function includes:

- To forbid the copy coming from other editor tools. That is the editor can prevent copy code from Internet or other electronic text. Our system forbids the code paste which coming from other editors. But the copy and paste is permitted in the anti-plagiarism editor.
- To prevent opening other code file. A student must login the system and editor the code, if they finished the assignment, they can submit the code to inductor. Every

students' code file attach with they own information, so if open the other's code file, they will can not normal display.

- Submit the code and collection into one folder: The students submit they code use the editor. The system will collect every assignment's code into one folder. It is convenient to detect details of every student's code and is convenient to detect other not-electronic plagiarism.
- Other kinds of plagiarism will be found plagiarism after submitting the code.

Figure 4 shows the process of submitting the source code after the student type the code. The system can collect the student code into a designated folder.

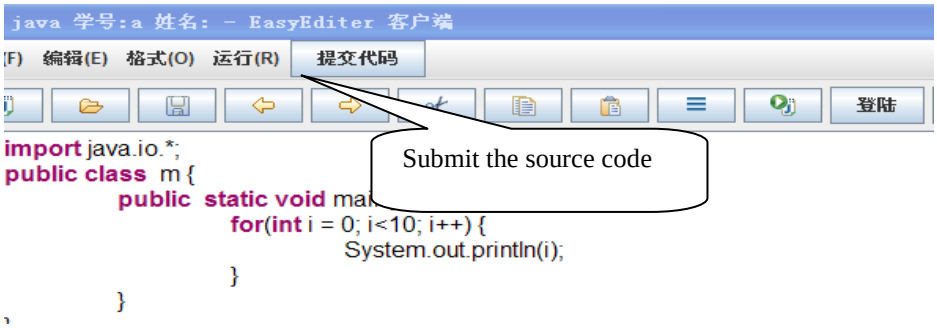

Figure 4. The Interface of the Editor System to Submit the Assignments

We have collected 38 student assignments. The number of code lines is about 70-100. The assignment is about how to type the calendar. We have analyzed these codes and get the rudimentary statistics about plagiarism. The Table 1 shows the details.

Table1. The Analysis before using Automatic Detection Tool

		N
electronic copy	**copy the whole code**	**8**
	copy the mainly code	**4**
modify	**modify base sentence of the code**	**4**
	modify structure but not change algorithm	**2**
	Non-plagiarism	**20**

3.2.2. Analysis the Student's Assignments using the Jplag System: JPlag is a system that finds pairs of similar programs among a given set of programs. We submit the source code files of the student's assignments to the JPlag. The web address is http://www.ipd.uni-karlsruhe.de/jplag. We have submitted the folder of above 38 student's assignments. The result shows in the Figure 5, after analyzing the student's assignments by using the Jplag system.

JPlag takes as input a set of programs, compares these programs pairwise (computing for each pair a total similarity value and a set of similarity regions), and provides as output a set of HTML pages that allow for exploring and understanding the similarities found in detail. Through looking over the results, we can check every higher similarity level's source codes.

The similarity value is a percentage. It's a value from 0%-100%, the higher of the value show that the similarity is greater.

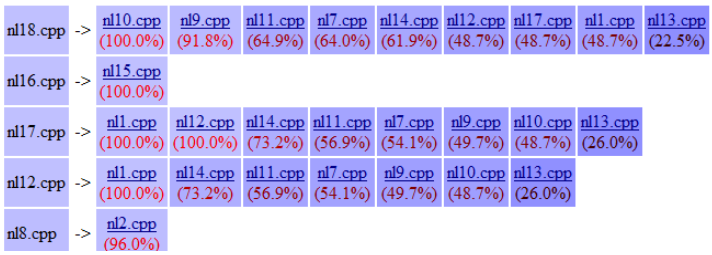

Figure 5. The Higher Similarity of Two Source Codes

An important question for plagiarism detection systems is the presentation of their results: Similarities of 0% or 5% can be represented by the similarity value alone — this clearly is no plagiarism. Likewise, similarities of 100% can also be represented by the similarity value alone — this clearly is a plagiarism. But what if the similarity is 40%? Such cases should usually be investigated by an inductor.

In this phase, we filter out similarities of 0%-10% which can be confirmed non-plagiarism. We also filter out similarities of 90%-100% which can be confirmed plagiarism. Other results need be investigated in next phase.

To above sets of 38 submitted assignments, the preliminary statistics result about plagiarism is shown in Table 2.

Table 2. The Preliminary Statistics Result about Plagiarism

	N
plagiarism	**10**
suspicion	**13**
Non-plagiarism	**15**

3.2.3. Confirmation the Suspicion after Analysis the Results of Jplag: Opening the hyperlink in the Figure 5, we can look up the details of every two higher similarity source code. Figure 6 show the similarity part signed with different colors. It is very convenient for instructor to look over the similarity details and to find adequately proof to confirm the plagiarism's suspicion.

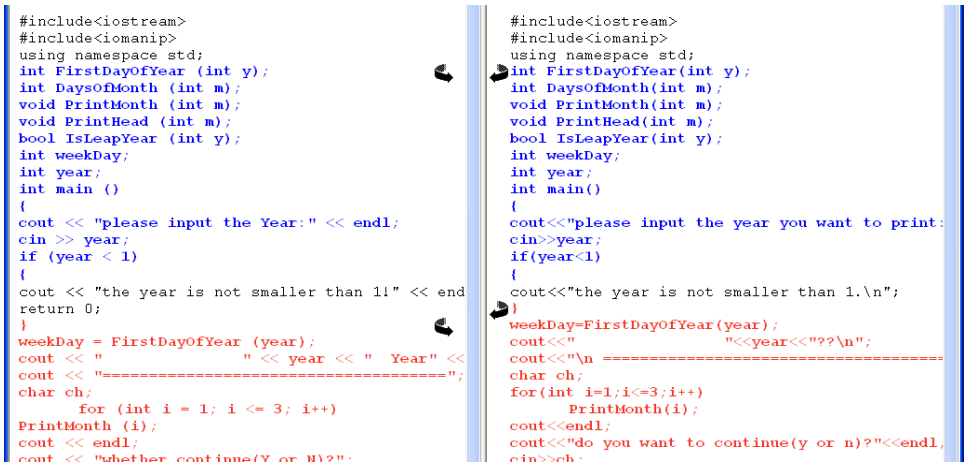

Figure 6. The Details of Two Higher Similarity Source Codes

We open every hyperlink of the similarity more than 10%, and look over the details of every pair similarity source codes. The aim is to find the suspicious. For each pair selected by the instructor, a side-by-side comparison of the programs is then shown (see Figure 6 for a partial example). Ranges of lines that have been found to be corresponding in both files are marked with the same color. It proves us enough proof to show that the students plagiarize other's work.

To above sets of 38 submitted assignments, the statistics result after being investigated by inductor is shown in Table 3.

Table 3. The Statistics Result after be Investigated by Inductor

	N
plagiarism	**10**
suspicion	**10**
Non-plagiarism	**18**

3.2.4. Investigation the Suspicions to Test and Verify the Plagiarism: We can not say with certainty that some students have plagiarism behavior only based on the higher similarity. The reasons include follow aspects:

- The student's assignment usually is uncomplicated and the achievement relatively fixed
- The student have the similar style and habit for the identical teacher and environment
- The student usually finds the similar method to solve the assignment. Because they are beginners to programming and theirs techniques are limited.

So we must test and verify plagiarism after finding suspicions. This process must finish by instructor, because the instructor understands his students. And know how to ask about the details of achievement the program to the suspicious. Combining the result of the above there phases, it is very easy to verify the plagiarism.

To above sets of 38 submitted assignments, the statistics result after investigate the suspicions by one's inductor is shown in Table 4.

Table 4. The Statistics Result after Investigate Suspicious

	N
plagiarism	**18**
Non-plagiarism	**20**

4. Conclusions

This paper provides a two-stage approach to reduce the incidence of plagiarism in software development courses. One is to prevent plagiarism from occurring, the other is to detect plagiarism when the preventative measures fail. During these two stages, we use the automatic tools to help the tutor prevent and detect plagiarism. One is the anti-plagiarism editor system to prevent the most common electronic copy plagiarism, in addition, the editor is convenient to collect the students programming assignments in a folder, the folder should submit the Jplag system which can automatically detect the plagiarism. This system is based on the Java, it has been completed by us. The other automatic tool is the Jplag system, this

system provides online server by Karlsruhe University. It can get the higher similarity programs and the details of every two similar part. This information is reference to detect plagiarism.

After using these processes, the tutor can faster and more accurately find the plagiarism suspicion. And using the automatic software is helpful to deter the students from plagiarism.

Acknowledgments

This research was supported by the Natural Scientific Research Project of Inner Mongolia Autonomous Region, China (2011MS0906).

References

[1] J. Harris, "Plagiarism in Computer Science Courses", Proceedings of the Conference on Ethics in the Computer Science Age, **(1994)**, pp. 133-135.
[2] N. Wagner, "Plagiarism by Student Programmers", http://www.cs.utsa.edu/~wagner/pubs/plagiarism0.html, **(2013)**.
[3] J. Sheard, "Cheating and Plagiarism: Perceptions and Practices of First Year IT Students", Proceedings of ITiCSE, **(2002)**, pp. 183-187.
[4] G. Whale, "Identification of program similarity in large populations", The Computer Journal, vol. 33, no. 2, **(1990)**.
[5] L. Prechelt, G. Malpohl and M. Philippsen, "Finding plagiarisms among a set of programs with JPlag", Univ. Comput. Sci., vol. 8, no. 11, **(2002)**, pp. 1016-1038.
[6] S. Schleimer, Wilkerson and A. Aiken, "Winnowing: local algorithms for document fingerprinting", SIGMOD ACM Proc. Int. Conf. Management of Data, San Diego, CA, ACM Press, NY, **(2003)** June, pp. 76-85.
[7] S. Mann and Z. Frew, "Similarity and originality in code: plagiarism and normal variation in student assignments", Conferences in Research in Practice in Information Technology, vol. 52, **(2006)**.
[8] P. Vamplew and J. Dermoudy, "An Anti-Plagiarism Editor for Software Development Courses", Research and Practice in Information Technology, vol. 42, **(2005)**.
[9] A. Parker and J. Hamblen, "Computer algorithms for plagiarism detection", IEEE Transactions on Education, vol. 32, no. 2, **(1989)**.
[10] K. J. Ottenstein, "An Algorithmic Approach to the Detection and Prevention of Plagiarism", ACM SIGCSE Bulletin, vol. 8, no. 4, **(1976)**, pp. 30-41.
[11] S. Grier, "A Tool that Detects Plagiarism in Pascal Programs", Twelfth SIGCSE Technical Symposium, St Louis, Missouri, **(1981)**, pp. 15-20.
[12] J. A. W. Faidhi and S. K. Robinson, "An Empirical Approach for Detecting Program Similarity within a University Programming Environment", Computers and Education, vol. 11, no. 1, **(1987)**, pp. 11-19.
[13] W. K. Joo, K. S. Choi, Y. J. Shin and J. S. Kim, "A Study of Design and Implementation of Korean Plagiarism Detection System", IJSEIA, vol. 7, no. 1, **(2013)**, pp. 211-220.
[14] C.-F. Wu, C.-T. Lin and P.-M. Wang, "Applying Multi-Criteria Method to the Decision of Assessment Tools for High-care Student Groups", IJHIT, vol. 6, no. 3, **(2013)**, pp. 1-14.
[15] H. Park, "Relationship between Motivation and Student's Activity on Educational Game", IJGDC, vol. 5, no. 1, **(2012)**, pp. 1-22.
[16] J. Zobel, "Uni cheats racket: a case study in plagiarism investigation", Proceedings of the sixth conference on Australian Computing Education, **(2004)**, pp. 357-365.

Author

Wang Chunhui received the BS and MS degrees in computer science from Inner Mongolia Normal University, China, in 2002 and 2008. Currently, her research interests include software analysis, multi-media and CAI.