# SURVEILLANCE IN PROGRAMMING PLAGIARISM BEYOND TECHNIQUES: AN INCENTIVE-BASED FISHBONE MODEL

Yanqing Wang, Min Chen, Yaowen Liang and Yu Jiang
*School of Management, Harbin Institute of Technology, Harbin 150001, China*

**ABSTRACT**

Lots of researches have showed that plagiarism becomes a severe problem in higher education around the world, especially in programming learning for its essence. Therefore, an effective strategy for plagiarism surveillance in program learning is much essential. Some literature focus on code similarity algorithm and the related tools can help to deal with the tedious and time-consuming task of detecting plagiarism to some extent. However, it is somewhat cursory to determine plagiarized programs only with automatic tool. Meanwhile, there is little research on incentive strategy of plagiarism surveillance. In this paper, we focus on the incentive mechanism of programming plagiarism surveillance. We aim to use some incentive strategy for plagiarism to gain the students' learning outcome. In the context of a programming learning information system EduPCR, a fishbone model is proposed, which consists of three aspects: problem, reason, and strategy. The questionnaire and interview conducted to survey the students' attitude about the incentive model show that the incentive model is welcome and practical in programming plagiarism surveillance. Our research is not only valuable for the programming learning but also for plagiarism surveillance of documents in other formats in academia.

**KEYWORDS**

Plagiarism surveillance, Incentive mechanism, Fishbone diagram, EduPCR

## 1. INTRODUCTION

Since teachers in academia often evaluate students' learning effect towards their assignments submitted. Plagiarism becomes a severe problem in higher learning around the word, especially in programming leaning since its code rules. As Parker put that if a code is all copied from another one or there is a little modification in it, then it can be considered as code plagiarism (Schleimer et al, 2003).

Some literature focus on code similarity algorithm and many software tools have been developed. For example, in the context of programming assignment, techniques for measuring code-similarity were adapted to detect plagiarism of assignments (Parker et al, 1989; Joy & Luck, 1999; Ji, 2007). GPLAG used program dependence graph analysis to identify code plagiarism (Liu et al, 2006). PlaGate can be integrated with existing plagiarism detection tools to improve plagiarism detection performance (Cosma & Joy, 2012).

Many survey-based researches were conducted to clarify the definition of plagiarism in programming assignments. Dick et al (2001) launched a scenario-based survey to discover the acceptability of different scenarios related to plagiarism, and reached a consensus on plagiarism behavior. Cosma and Joy (2008) used a set of scenarios to collect opinions from academics, and concluded a definition of source code plagiarism. Bennett et al (2005) built a model to explain the incidence of plagiarism and tested it through a survey collected from 249 students. To explore university students' perceptions of plagiarism, a focus group study is proposed (Gullifer & Tyson, 2010). To support deep understanding of different linguistic patterns, a new taxonomy of plagiarism was presented (Salha et al, 2012).

There are a lot researches on anti-plagiarism. Walker et al (2010) did research on measuring plagiarism. Shen et al (2007) proposed a flexible anti-plagiarism system model based on user-defined plagiarism standards. Ranald et al (2006) presented a checklist for identifying the absence of a holistic approach to dealing with student plagiarism. The attitude and strategies about plagiarism and why it happens should also deserve attention. Park (2004) carried on a research to explore why an institutional framework for dealing with plagiarism by students is necessary. To study investigates factors affecting the adoption of anti-

plagiarism software, Lee (2011) used protection incentive theory as a basis, examined influence of threat and coping appraisals.

Since anti-plagiarism techniques are relatively mature, we believe that the ideal solution to plagiarism should base on more management than technique. In this paper, we design a fishbone diagram model for plagiarism surveillance based on incentive mechanism. After all, there needs to be an incentive for students to learn more (Dana V., 2011). The fishbone model is embedded in an e-learning system EduPCR.

## 2. BACKGROUND

Code review was firstly introduced by Fagan in 1976, and then has been a common programming practice and widely used for many years (Fagan, 1976; Fallows & Chandramohan, 2001; Mäntylä et al, 2009). Afterwards observing the successful practices of code review in software industry, many computer science educators have shown their interests in introducing peer code review (PCR) processes to their courses and a number of good learning outcomes have been reported (Li, 2007).

We designed and implemented a PCR system EduPCR and applied it in the assessment of two programming language courses *C programming* and *Object Oriented Programming in Java*, and works very well to improve learning outcome (Wang et al, 2012). The activity diagram of EduPCR is depicted in Fig. 1.

**(1) New assignment**. In this phase, the teacher sets a new assignment to the system, and the system will inform all students on class though short massages which consist of both job description and deadline.

**(2) Submit manuscript**. After receiving the notification of new assignment, students should finish their manuscripts (first edition of code) and submit them as quickly as possible. Then the computer scans the number of students who submit their manuscript in a certain interval. When the number is more than or equal to 3, the reviewer assignment is performed among them. When only one or two has not been assigned, they will be merged into the just-built-group (more than or equal to 3 members) and assigned among them.

**(3) Review**. Student who is assigned as reviewer will receive a short massage as notification. Afterwards, they should finish their review task and submit comments before the deadline.

**(4) Revise manuscript**. After a reviewer finishes his/her review task and submits comments, the author will receive a notification that he/she should revise their manuscript according to reviewer's comments and submit the revision code before the deadline.

**(5) Check marks**. When all students' revisions have been submitted or the whole assignment is done, the teacher will give students marks for their work concerning the qualities of manuscript, review and revision. Then, students can check their marks on the system.

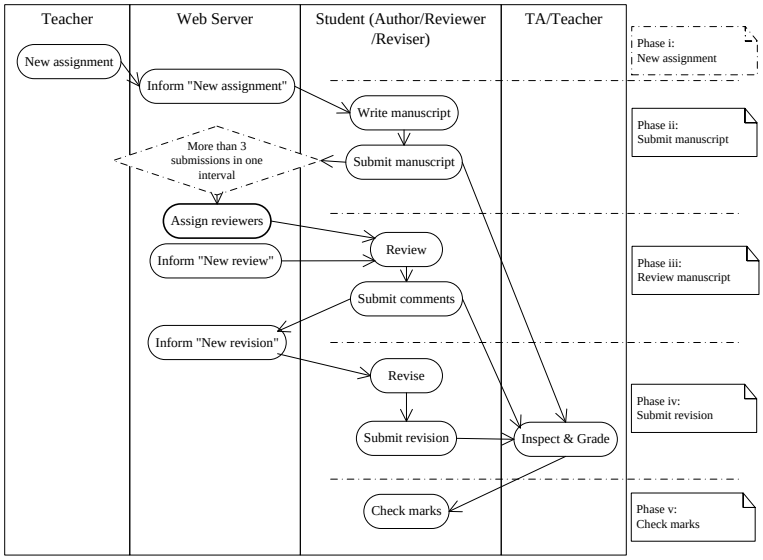


Figure 1. Activity diagram of EduPCR

# 3. PLAGIARISM SURVEILLANCE MODEL IN FISHBONE DIAGRAM

Our plagiarism surveillance model is in PRS (problem, reason and strategy) infrastructure and has 6 sub questions including *resource*, *individual difference*, *source code*, *penalty by teacher*, *schedule*, and *learning outcome*, as described in Fig. 2.

## 3.1 Resources

Students can get the code resource from many ways to complete a task, such as Internet, open source software, and students who have completed the task. In our opinion, the third case, just mentioned, is probably of plagiarism. To minimize the influence of plagiarism, we utilize a filter strategy. It means that when a student submits his/her source code, in *phase i* of Fig. 1, the system will compare his/her code with all code submitted by other students. If the code similarity exceeds a pre-defined threshold (we will adjust it in practical situation), the system will reject this submission. He/she can apply teacher's arbitration. Teacher's reflection options include *rejecting application*, *receiving application*, or *increase the thresholds* (after a certain number of arbitrations).

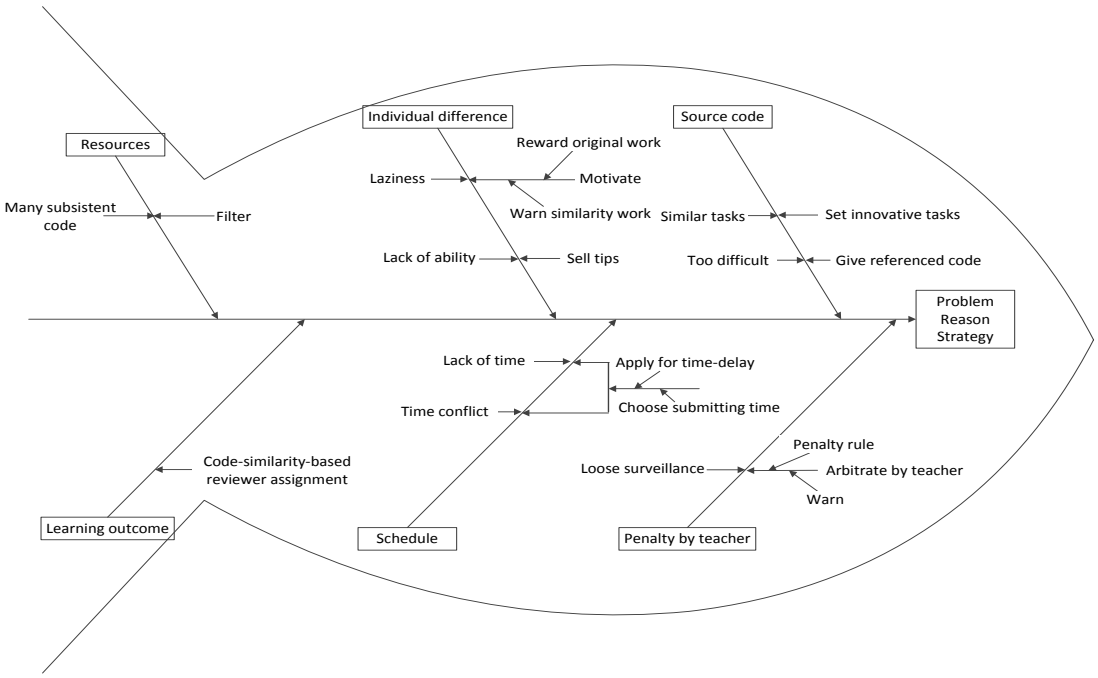


Figure 2. Fishbone model of plagiarism surveillance

## 3.2 Individual Difference

It consists of laziness and lack of ability. Actually, not all students are diligent, and some students are too lazy to complete the task. They tend to plagiarize others' works. Similarly, some students are lack of programming ability to complete a task independently, they have to refer to the others' works.

To motivate the lazy students, the students with original work will be rewarded while those with similar code will be warned, as depicted in Fig. 2 and the *phase v* of Fig. 1. The system displays the top ten highest similar works and the top ten lowest similar ones in different sections. For the students who are lack of ability to complete a task, we will prepare some tips. A tip can be a code segment or algorithm description, as described in Fig. 2 and the *phase ii* of Fig. 1. At each time when a student utilizes a tip, he/she will get a small-amount deduction of mark.

## 3.3 Source Code

The tasks of programming course maybe relatively similar. Students can easily find the same tasks from Internet or some higher grade students which can be regarded as plagiarism. Therefore, teachers should try hard to set innovative tasks to reduce the similar tasks, as described in Fig. 2 and the *phase i* of Fig. 1. Besides, some tasks are so difficult that every students have to plagiarize by various ways. To decrease the difficulty of the task as well as the number of students who plagiarize for this reason, teachers are asked to give some referenced code before a very difficult assignment is starting *phase i* of Fig. 1.

## 3.4 Penalty by Teacher

Teachers' loose surveillance is another reason leads to plagiarism so that we set a penalty rule uniting teachers' arbitration as described in Fig. 2 and the *phase v* of Fig. 1. That is, once we find plagiarism happens between two students, we penalize both of them. Each of them can ask for teacher's arbitration and retexture. After the arbitration, if one didn't plagiarize or one is plagiarized by the other unwittingly, he/she will not get penalty, while the one who plagiarized will get a more serious penalty.

## 3.5 Schedule

Students in high education always feel lack or conflict of time, as it described in Fig. 2. And for the schedule factors, plagiarism problem obviously increase according to the interview results. To this, we design a strategy that students can choose time to submit their manuscript and apply for time-delay. But without doubt, the options and application should obey some regulation which is showed as below.

We set four options for submitting deadline, as it presented in Fig. 3. Teacher should make option seriously considering both the course arrangement and students' other exams. And in order to incentive students to submit manuscript early, if one student chooses to submit his manuscript (e.g. option 1) before the teacher's setting (e.g. option 2) and does submit his program on schedule, he/she will get an award in score. In contrast, if one chooses the time (e.g. option 4) much later than the teacher's setting (e.g. option 2), he/she will get some deduction in score. Considering about someone's particular situation, those who choose the time (e.g. option 3) won't get any loss in score.

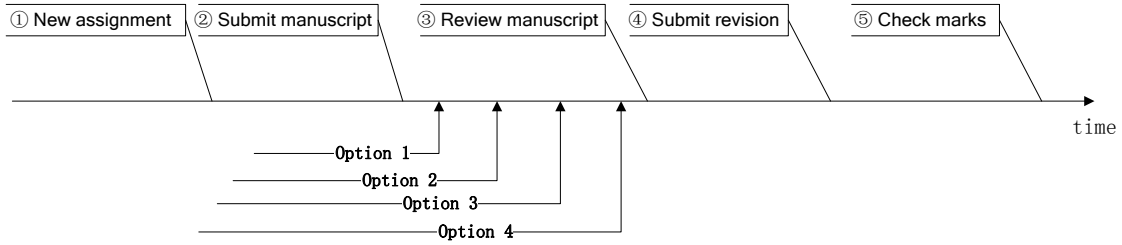


Figure 3. Submission time selection and time-delay request

## 3.6 Learning Outcome

It's obvious that plagiarism problem will severely reduce learning outcome, and it is inevitable though we take a series of measures and strategies for plagiarism surveillance. Concerning the ultimate goal of education is helping students to learn more, we propose an code-similarity-based reviewer assignment strategy, as described in Fig. 2 and the *phase ii* of Fig. 1. It can help students improve learning outcome even they have plagiarized. In this approach, students who plagiarise or have similar programs will be assigned in the same group. In order to improve their programming skill and learning skill, a reviewer should submit not only their comments but a report, as depicted in the *phase iii* of Fig. 1. The report describes what is similar in their programs such as program structure, algorism etc. In order to get a high grade, students have to write the report seriously, and their programming and learning skill will be improved to some extent.

# 4. INVESTIGATION

## 4.1 Questionnaire

To validite the plagiarism surveillance model, a survey is conducted to the students who used EduPCR.

Firstly, we investigate how many times have the student plagiarized in four aspects, and the results, presented in Table 1, reveal that the plagiarism problem is very serious. In the table, the item "Request" presents the times of requesting to plagiarize other people's assignment. The data shows nearly all of the students have plagiarized others' programming works and more than half of them have plagiarized more than 4 times. The next item "Offering" means the times of offering assignment to good friends. According to the statistical result, more than half of the students have offered their homework to close classmates. The following item "Algorithm" stands for the times of referring to the major algorithm of others. And nearly 70% of the students admitted that they have referred to the mainly algorithm of others. The last item "Similar" indicate the times of searching and referring to the similar answers from the Internet, and almost half of the students have referred to or even just copy the answer to the similar task from the Internet.

Table 1. The results of Q1 in questionnaire

| Times | 0 | 1 to 4 | More than 4 | Total |
|---|---|---|---|---|
| Request | 0.033 | 0.396 | 0.571 | 1.0 |
| Offering | 0.483 | 0.407 | 0.11 | 1.0 |
| Algorithm | 0.352 | 0.571 | 0.077 | 1.0 |
| Similar | 0.516 | 0.363 | 0.121 | 1.0 |

After realizing the serious plagiarism in code programming, we investigate the students' attitude towards the planning to carry out plagiarism surveillance in the EduPCR system. The results show that 69.15% of students agree with the idea of plagiarism surveillance, which indicates that the idea of plagiarism is valid.

In order to prove the plagiarism surveillance model is successful, we have set questions about some strategies in the plagiarism model. For example, we set filter with a certain threshold (such as 85%) in the system, when someone submits his or her homework, the system will compare his/her homework with the submitted programs of other students, and if the similarity exceeds the threshold, the students cannot submit his/her homework successfully. According to investigation, only 41.3% of students think the strategy of setting a certain threshold is reasonable, and more than half of the students think it is not reasonable for the reason that the threshold should adjust to different task.

Another example is about teacher's arbitration to the filter and the penalty rule mentioned above. About 52.2% of the students think the strategy is reasonable; 33.7% of the students think it is just ok; and only 14.1% of the students think it is not very reasonable. And we can conclusion that the strategy of teacher's arbitration to the filter and the penalty rule is much accepted.

## 4.2 Interview

To comprehend the plagiarism in programming better and ameliorate our surveillance model, we have interviewed some students who have finished the class of C or Java. The record can be concluded as follows. (1) The reasons of the plagiarism could be lack of time and programming ability. The conflicts between programming with other things will also lead to plagiarism. (2) The methods students most take to plagiarism include copying or referring to others and searching on internet. (3) Most students think if there are some strategies to surveille the plagiarism, they can refuse close classmates' request to copying program without the risk of breaking up. And some students think when the task is very difficult, there should be some tips otherwise the strategy may take much inconvenience to them. All in all, the reasons and methods of plagiarism are various, and the surveillance for plagiarism is much necessary.

# 5. CONCLUSION

Plagiarism problem is an old problem in high education especially in programming study, and there have already some measures to prevent it. However, there is not a satisfactory effect yet because plagiarism is inevitable among students. Considering about its essence of inevitability, we propose a plagiarism surveillance strategy by designing a fishbone model based on incentive mechanism and introduce it into our EduPCR system. According to questionnaire and interview among students who have used the system EduPCR, this model is reasonable and practical. Especially, to promote the overall learning outcome, this model is based on incentive mechanism. In a word, we regard the plagiarism problem as more management issue than technical one. Nevertheless, there are still some issues remaining to discuss.

*Should the model be technique-based or management-based?* As it's described above, our research on plagiarism surveillance is based on incentive mechanism which belongs to a management domain. However, strategies such as code-similarity-based reviewer assignment, the measurement of code similarity are indeed about technique. So our plagiarism surveillance model is technique-based at first stage, then it should be management-based more and more.

*Whether the new model could be introduced directly into another existing e-learning system?* A good model should be flexible. Since our plagiarism is based on the e-learning system developed by ourselves, the need to implement another e-learning system could be a challenge. Someone may wonder whether this model could be introduced directly into another existing system. That is a requirement to increase the model's adoption of other e-learning system and is what we should think it over. Fortunately, most of our strategies in the fishbone model can apply to other e-learning system. To the strategies dedicated to EduPCR system, we will improve their adoption to others system by more precise design.

In the future, we will conduct a case study of 4 years students' programming data on C and Java programming courses to find their plagiarism status or tendency. Thus we can improve our plagiarism surveillance model more flexible and more effective according to the case study. The investigation can help us recognize students' plagiarism behavior better as well.

# REFERENCES

Alzahrani, S. M., Salim, N. and Abraham, A., 2012. Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, Vol. 42, No. 2, pp 133-149.

Bennett, R., 2005. Factors Associated with Student Plagiarism in a Post-1992 University. *Assessment and Evaluation in Higher Education*, No. 30, pp 137-162.

Cosma, G. and Joy, M., 2008. Towards a Definition of Source-Code Plagiarism. *IEEE Transactions on Education*, Vol. 51, No. 2, pp 195- 200.

Cosma, G. and Joy, M., 2012. An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis. *IEEE Transactions on Computers*, Vol. 61, No. 3, pp 379-394.

Dick, M., Sheard, J., Markham, S., 2001. Is It Okay to Cheat? - the Views of Postgraduate Students. *ACM SIGCSE Bulletin*, Vol. 33, No. 3, pp 61-64.

Fagan, M. E., 1976. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, Vol. 15, No. 3, pp 182-211.

Fallows, S. and Chandramohan, B., 2001. Multiple approaches to assessment: aeflections on use of tutor, peer and self-assessment. *Teaching in Higher Education*, Vol. 6, No.2, pp 229-246.

Gullifer, J. & Tyson, G. A. 2010. Exploring university students' perceptions of plagiarism: a focus group study. *Studies in Higher Education*, Vol. 35, No.4, pp 463-481.

Ji, J. H., Woo, G. and Cho, H. G., 2007. A source code linearization technique for detecting plagiarized programs. *ACM SIGCSE Bulletin*, Vol. 39, No. 3, pp 73-77.

Joy, M. and Luck, M., 1999. Plagiarism in Programming Assignments. *IEEE Transactions on Education*, Vol. 42, No. 2, pp 129-133.

Lee, Y., 2011. Understanding anti-plagiarism software adoption: an extende protection motivation theory perspective. *Decision Support Systems*, Vol. 50, No. 2, pp 361-369.

Li, X., 2007. Incorporating a code review process into the assessment. Proceedings of the 20th Annual Conference of the National Advisory Committee on Computing Qualifications. Nelson, New Zealand, pp 125-131.

Liu, C., Chen, C., Han, J., and Yu, P. S., 2006. GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis, in: KDD, pp 872-881.

Mäntylä, M. V. and Lassenius, C., 2009. What types of defects are really discovered in code reviews? *IEEE Transactions on Software Engineering*, Vol. 35, No. 3, pp 430-448.

Park, C., 2004. Rebels without a clause: towards an institutional framewoek for dealing with plagiarism by students. *Jounal of Further and Higher Education*, Vol. 28, No. 3, pp 291-306.

Parker, A. and Hamblen, J. O., 1989. Computer Algorithms for Plagiarism Detection. *IEEE Transactions on Education*, Vol. 32, No. 2, pp 94-99.

Schleimer, S., Wilkerson, D. S. and Aiken, A., 2003. Winnowing: local algorithms for document fingerprinting. Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. ACM, pp 76-85.

Shen, Y., Yuan Z., Liu L. and Dong H., 2007. Research of Anti-Plagiarism Monitoring System Model. *Wuhan University Journal of Natural Sciences*, Vol. 12, No. 5, pp 937-940.

Vedder-Weiss, D. and Fortus, D. 2011. Adolescents' Declining Motivation to Learn Science: Inevitable or Not. J*ounal of Research in Science Teaching*, Vol. 48, No. 2, pp 199-216.

Wang, Y., Li, H. Feng, Y., Jiang, Y. and Liu, Y., 2012. Assessment of programming language learning based on peer code review model. *Computers & Education*, Vol. 59, No. 2, pp 412-422.