5th International Conference on Higher Education Advances (HEAd'19)
Universitat Politècnica de València, València, 2019
DOI: http://dx.doi.org/10.4995/HEAd19.2019.9297

Effects of Plagiarism in Introductory Programming Courses on the Learning Outcomes

Dieter Pawelczak
Institute for Software Engineering, Bundeswehr University Munich, Germany.

Abstract

We compare two introductory programming courses and the accompanying programming assignments with respect to the learning outcomes and the relation to plagiarism. While in the first course the solutions from the students of their programming assignments are checked directly with a plagiarism detection system to prevent students from plagiarizing, plagiarism is not tracked in the second course. Running a post check against plagiarism after the course reveals a significant higher plagiarism rate with several exact copies. As the number of students handing in copies from fellow students increases, the failure rate in the final examination also rises. Analyzing the data does not only reveal a correlation between plagiarizing and inferior examination results, but also shows, that students confronted with a plagiarism detection system have better skills in fundamental coding concepts. We suppose this might be a result of the fact, that the implementation of a plagiarism detection system does not deter so many students from plagiarizing, but students are strongly motivated to run more modifications on their plagiarisms in order not to be caught.

Keywords: *Plagiarism; Source code plagiarism; Teaching programming; Automated assessment systems.*

1. Introduction

Many undergraduate students in engineering degree programmes have difficulties with learning programming. The curriculum usually involves practical exercises and/ or programming assignments in order to train programming skills, as learning programming requires a significant personal engagement to understand and learn to apply fundamental programming concepts. A student evaluation for such courses typically reflects the high workload required for the practical work. Unfortunaltely, a common approach to lower the effort is to use or adapt solutions of fellow students. Universities implement source code plagiarism detection systems in order to detect and prevent such frauds, compare e.g. Modiba et al. (2016). We were running such a system for 5 years and made experience of many positive aspects, but also found some draw backs, especially with respect to the social behavior of the students as described by Pawelczak (2018). In order to further investigate the effects of plagiarism, we disabled the plagiarism detection system in 2018 and compared the learning outcomes of the course (hereinafter called class B) with the previous year (refered to as class A). We communicated to the students of class B, that plagiarizing would not be tracked during the course. We also made clear, that our experience showed, that plagiarizing has a negative effect on passing the course. Nevertheless, we expected a higher plagiarism rate and also a weaker performance in the course examination. Running the plagiarism detection system after the completion of the course by comparing all solutions of the students with each other, revealed a significantly higher plagiarism rate. Furthermore, the failure rate in the examination rose by 17 %, which supports our assumption, that plagiarizing has a negative effect on the learning outcomes. During the course and while analyzing the performance of the individual students, we found other interesting details, which are discussed in the following chapters.

2. Related Work

We define source code plagiarism as discussed by Cosma and Joy (2008) as reproduction/ copying source-code either without making any adaptions or just providing moderate alternations. Students violate the academic integrity by pretending to be the author of another one's work. Especially in programming courses this is a wide spread phenomenom, as it is very easy to copy a working solution. According to Fraser (2014) reasons for cheating are the lack of interest in the task, insufficient skills or time pressure. Some students also think, that working on the task has no benefit for them. Additionally, as students often work in groups, they do not see a violation, if they all hand in the same solution. As Joy et al. (2013) describe in their study on the students perspective on source code plagiarism, that universities usually implement plagiarism detection systems to deter students from plagiarizing. However, there is often a different understanding of what plagiarism means, e.g., if the lecturer provides code snippets in the lecture – are students

allowed to re-use them in their programming assignments? Students also learn, that re-using code is a paradigm of object oriented programming. Why should their programming work at the university differ from the real world? Two comprehensive studies from Joy et al. (2013) and Simon et al. (2018) emphasize the need for the lecturer to spend more time on educational work with respect to plagiarism and to use a transparent policy when pursuing plagiarism.

Palazzo et al. (2010) showed the correlation between plagiarizing and the learning outcomes in physics education. Altough some students cheat, because they already accomplished the required skills, for the average students, cheating results in less effort spent on the course subject and typically in poorer examination performance. Their study confirms our experience, that it is not sufficient to inform students of the negative effects of cheating, as this will not reduce it. Therefore, a proper stategy to handle plagiarizing is required in order to reduce plagiarizing.

As Bradley (2016) states, source codes in introductory courses provide a high natural similarity, as students are taught to code with a particular coding standard, or as students might be required to use the same names for functions and variables for an automatic grading systems, or because of the use of code snippets from the text book. Bradley suggests to use a randomization of tasks in order to increase the differences among the students's solutions. For our course, we use the tool PlagC2 for the plagiarism detection, which allows common code snippets to be removed. Due to the fact, that most submissions have only around 140 lines of code the natural similarity is typically between 60 and 80 %. Common parts with respect to the programming assignments are taken out before the comparison, e.g. given function prototypes or example code snippets shown in the lecture are removed in order to focus on the students' independent work.

3. Data and Methodology

The introductory C-programming course for first year engineering students requires students to submit seven programming assignments in digital form, and to pass the final written examination.

3.1. Data

The data for the analysis comes on the one hand from the submitted programming assignments of the last two years and on the other hand from the examinations and the students' evaluation of the course. 50 students attended class A and 51 class B. In each course, student feedback is requested. For both classes, the response rate was about 64 %. To analyse plagiarizing among students, about 350 source texts per course are available. We can also access the submission statistics of the automated assessment system, which tracks information on all submissions, e.g. time stamps, incomplete or erroneous submissions as well as detected plagiarism.

3.2. Plagiarism detection

Each source code a student submits, is stored in a database. During submissions, the *PlagC2* tool calculates the similarity of that source code with the sources in the database and returns the highest similarity together with the ID of the matching source code, compare Pawelczak (2018). The automated assignment system rejects submissions in case the similarity exceeds a given threshold. The students are allowed to re-submit another version, although the number of re-submissions is restricted. The threshold varys as the natural similarities are typically different depending on the programming assignment. The threshold is set sufficiently high in order to prevent false positives. In case the similarity exceeds the threshold, we blame the submitting student for plagiarizing.

For class A the submitted source codes were directly analysed by the plagiarism detection system during the submission. Although the system is not able to distinguish between author and plagiarist, as the detection relies on the time of submission, it is very easy for the course instructors to find out, who plagiarized: asking questions on implementation details reveals very easily the cheater. In class B the submissions were only checked against functional correctness without plagiarism detection. For the analysis, we simulated the automated assessment system and fed the system with the submitted source codes from the students in random order and tracked the similarity using the same thresholds.

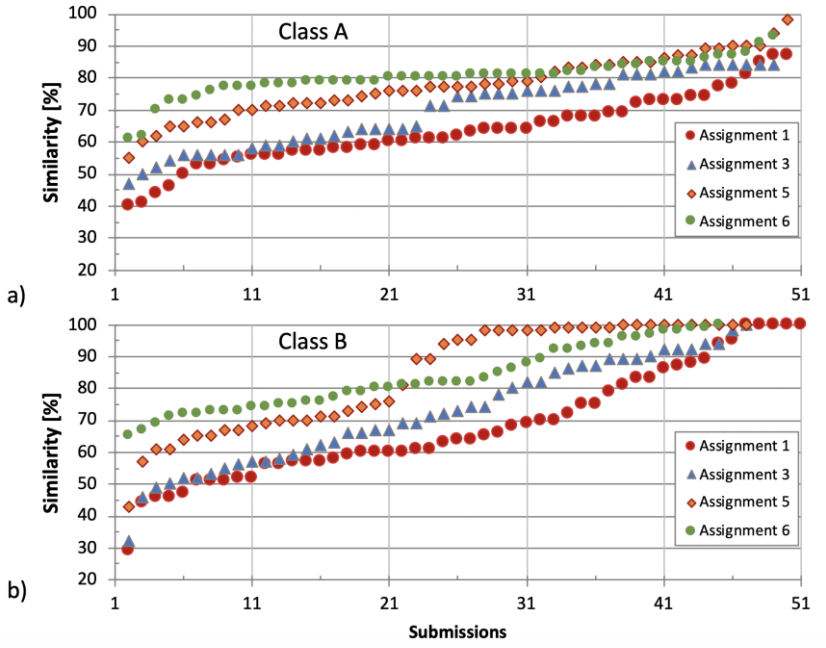

Figure 1. Source codes similarity for the programming assignments of class A(a) and class B(b)

4. Results and Discussion

4.1. Code Similarity and Plagiarism

Fig. 1 shows the calculated maximum similarity of each submitted source code exemplary for the programming assignments 1, 3, 5 and 6 for class A and B, respectively. For a better readability, not all assignments are printed in the diagrams and the similarity is listed in ascending order (not in the order of submission). As the data from class A already passed the plagiarism checks, despite some minor exceptions, the similarities are below the threshold. Table 1 shows the applied threshold and reveals the percentage of sources accused of plagiarism for both classes. As the effort increases with the later programming assignments, the percentage of plagiarizing increases in both classes. The average percentage of students plagiarizing in class A is 11.1 %, while it is 24 % higher in class B: on average 35.1 % of the students were caught plagiarizing with a maximum of more than 50 % in the assignments 5 and 6. On average class B provided 3.7 exact copies per assignment with a maximum of 10 in assignment 5.

Table 1. Results of the plagiarism detection analysis of the programming assignments

Assignments	**Integer**	**Floating Point**	**Strings**	**Structures**	**Arrays**	**Lists**	**Files**	**Average**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
Threshold	84 %	88 %	84 %	83 %	82 %	81 %	88 %	84.3 %
Average number of source lines	107	125	162	133	166	172	157	146
Average similarity in class A	64 %	76 %	69 %	75 %	78 %	80 %	80 %	74.7 %
Average similarity in class B	68 %	79 %	72 %	77 %	84 %	83 %	83 %	78.0 %
Percentage of students suspected of plagiarizing in class A	4 %	2 %	8 %	22 %	18 %	16 %	8 %	11.1 %
Detected submissions above threshold in class B	21 %	20 %	31 %	35 %	51 %	53 %	35 %	35.1 %
Unmodified 1:1 submissions in class B	4	3	1	2	10	2	4	3.7

4.2. Examination Results

There are many different influences on the examination results, which makes it difficult to directly compare the results of two courses. Therefore, we also compared the examination outcome of all other subjects for both classes and found on average a difference below 2 %. For instance, class A performed slightly better in *math* and *electrical science*, while class B performed slightly better in *computer science introduction*. From the overall performance of class B, we would have expected similar examiniation results compared to class A. Fig. 2 shows the examination results of both classes and reveals the distinct higher failure rate of class B. Although the plagiarism detection system can not distinguish between author and plagiarist, as discussed in Section 3.2, the students in class A, that were caught plagiarizing reached on average only 47 % in their exams and 8 from 17 students failed. One student in that group had an excellent exam. Although the sources were randomly submitted for class B, the system identified 34 students plagiarizing with 24 out of them failed the examination (with an average of 43 %). Many students among these 24 students were caught plagiarizing multiple times. The system detected a total of 111 sources in class B with the similarity above the threshold.

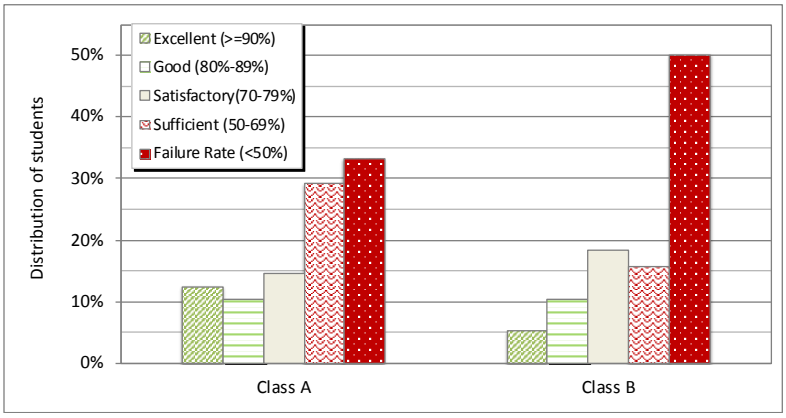

Figure 2. Distribution of students according to the examination results of class A & B.

Looking closer at the key competences students acquired in the examination, it was obvious, that students of class A had better skills in coding common programming constructs, like loops or functions and knew better how to use the standard library functions. We assume, that students do not plagiarize less, when the plagiarism detection system is active, but students have to take care, not to be caught. As the system detects lexical changes, students have to re-write the source code, they obtained from their fellow students. This improves coding skills. With respect to other programming skills like problem solving, we did not find remarkable differences in both classes.

4.3. Examination Results in Other Subjects in that Semester

We found an interesting aspect, when comparing the examination results of both classes in the corresponding semester: class A performed less compared to class B in *physics* and *measurement technology*. In the fourth course *electronic components* in that semester, class A performed slightly better. In average, class B performed 2.5 % better in the examinations concurrent to the programming course. If we take the programming course into account, class B performed 0.7 % worse. Although these tiny swings might be random, it might point to the fact, that course B had more time to prepare the other examinations as plagiarizing is less time consuming, compare Section 4.4.

4.4. Course Evaluation

Class A stated in the course evaluation with 228 hours a higher workload in average compared to class B with a workload of 195 hours. Fig. 3 shows an excerpt from the evaluation results. We asked, which means students found helpful to solve their programming assignments. There was less discussion among the students in class A: 63 % agreed, that discussion was helpful, while in class B 80 % agreed on that (Fig. 3, Question 1). We noticed, that some students in class A were not willing to share their ideas, because they feared, that they might be accused of plagiarizing. While 28 % of class A disagreed, that using code snippets or solutions from other students was helpful, in class B only 15 % disagreed with that. In class B a direct usage of these solutions was possible (Fig. 3, Question 2).

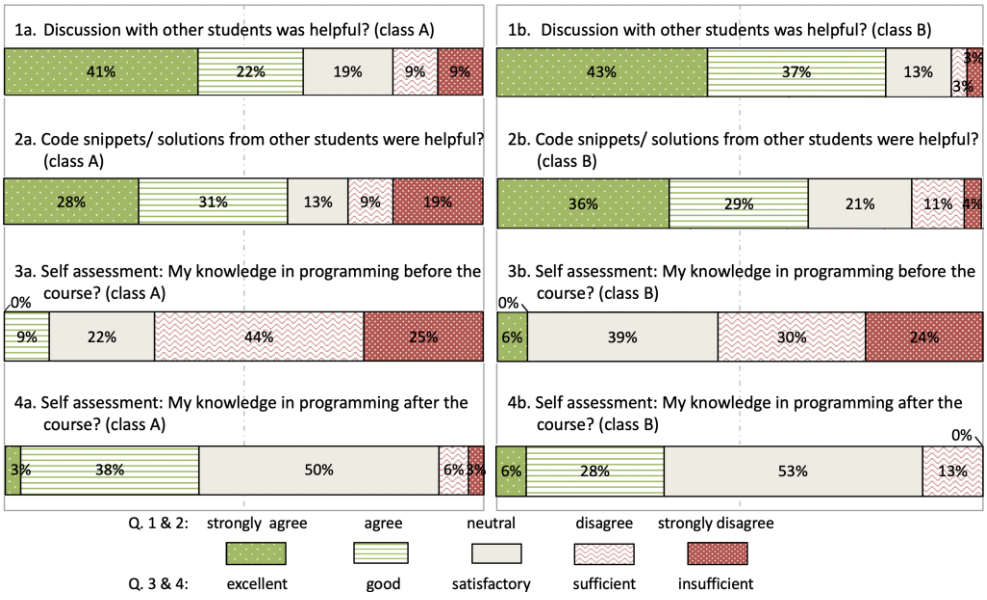

Figure 3. Excerpt from the students' evaluation of class A & class B.

Interesting is the comparison on the self-assessment of the students (Fig. 3, Questions 3 and 4). We ask about their knowledge of programming before and after the course. In both classes the majority judged their knowledge before the course as below satisfactory, more than 80 % felt their knowledge satisfactory, good and excellent after the course. In class A, a much higher development can be seen: In class A some students increased their assessment over 3 grades, i.e. from insufficient to good or sufficient to excellent, while in class B the difference was two grades maximum.

5. Conclusion and Outlook

The use of a plagiarism detection system does not prevent students from plagarizing. It has a deterrent effect on some students, and as we observed, sometimes even a disquieting one. It also challenges smart students to outwit the system. A large group of students will still use solutions from other students, but they have to spend time on modifying the solutions in order not to be caught plagiarizing. The last aspect especially has an effect on the learning outcomes. Students faced with the plagiarism detection system showed a better knowledge of fundamental coding skills like writing loops, making code more modular (e.g. by outsourcing code into functions), or finding alternative solutions (e.g. by using different API functions).

Plagiarism detection systems do not surpress open discussions and collaborations among students as the course evaluation revealed, but an effect of exclusion and reluctance is observable. As Fraser (2014) concludes, we need to establish a learner's friendly environment, which allows collaboration. We still need means to deter students from plagiarizing and to help them to invest more in their own work. We definitely will apply the plagiarism detection system in the upcoming course, but we also want to reduce the effort of the assignments 5 & 6 by adding more voluntary tasks, to see if this results in less students copying their solutions.

References

Bradley, S. (2016). Managing plagiarism in programming assignments with blended assessment and randomisation. In Proc. of the 16th Koli Calling Int. Conf. on Computing Education Research (Koli Calling '16), Koli, Finland, November 24-27, 2016, 21-30

Cosma, G., & Joy, M. (2008). Towards a Definition of Source-Code Plagiarism. IEEE Trans. Education, 51 (2), 195-200. doi: 10.1109/TE.2007.906776

Fraser, R. (2014). Collaboration, collusion and plagiarism in computer science coursework. Informatics in Education, 13 (2), 179-195

Joy, M. S., Sinclair, J. E., Boyatt, R. Yau, JY-K., & Cosma, G. (2013). Student perspectives on source-code plagiarism. Int. Journal for Educational Integrity, 9 (1).

Modiba, P., Pieterse, V., & Haskins, B. (2016). Evaluating plagiarism detection software for introductory programming assignments. Proc. of the Comp. Sci. Education Research Conf. 2016 (CSERC '16), 37-46.

Palazzo, D. J., Lee, Y.-J., Warnakulasooriya, R., & Pritchard, D. E. (2010). Patterns, correlates, and reduction of homework copying. Phys. Rev. ST Phys. Educ. Res., 6 (1)

Pawelczak, D. (2018): Benefits and drawbacks of source code plagiarism detection in engineering education. In Proc. of *IEEE Global Engineering Education Conf. (EDUCON'18)*, Tenerife, 2018, 1048-1056. doi: 10.1109/EDUCON.2018.8363346

Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A., & Sinclair, J. (2018). Informing students about academic integrity in programming. Proc. of the 20th Australasian Computing Education Conference (ACE '18). 113-122. doi: 10.1145/3160489.3160502