EPiC Series in Computing

Volume 69, 2020, Pages 197–205

Proceedings of 35th International Conference on Computers and Their Applications

New Approach to Teaching Computer Programming to Freshmen

Jacob Sukhodolsky

Saint Louis University, St. Louis, Missouri

sukhodpj@slu.edu

Abstract

Freshmen who take an introductory computer programming course often ask their classmates for help. In some cases, they even copy each other's programs. That is being considered as cheating. The problem of cheating in Computer Science students' homework assignments so far has been handled mainly through administrative punishment of the cheaters. The success of such an approach depends to a large degree on the ability of the instructor to recognize the fact of cheating, which is a complicated task. With a large number of students taking the course, identifying the cheaters sometimes requires considerable time. The author of this paper suggests a way of solving the cheating problem by encouraging students' cooperation rather than trying to fight it. He also suggests the way of changing the course grading policy emphasizes the importance of regular checking the students' understanding of the course material.

1 Introduction

Cheating on computer science assignments has been a major concern for a large number of universities, such as U. of Florida, Texas San Antonio, Princeton, Stanford, Carnegie Mellon, and MIT, for over twenty years [1], [2], [3], [4], [5]. Many instructors believe that cheating is so popular among computer science students because of the high frustration levels students experience trying to get a program to run [3]. The latter makes writing and testing a program a tedious task, which may require a considerable amount of students' time. Copying and editing someone else's program saves students' time they can devote to their other courses.

The strategy many universities employ in their fight against cheating in computer science assignment consists of two steps:

1. Detection of cheating
2. Dealing with cheating incidents

The simplest way to check if a student has cheated is to compare his/her program to those of other students. Finding numerous similarities between programs is a sign that cheating has taken place. The

described approach has serious weaknesses. For instance, if a programming assignment is relatively simple, and all the students of the class use the textbook example of a program similar to the one assigned, there is a strong possibility that a number of programs written independently will look similar to each other. Another problem is a large size of the class, which requires the instructor to spend a prohibitive amount of time comparing students' code. Some educators recommend using software, like Moss [3], [4], which detects program similarities. Good results have been obtained with Caveon, software that exposes cheating by using statistical anomalies [4]. Another statistical method used to determine if cheating has taken place is described in [7].

If a student is identified as a cheater, the instructor has to choose the appropriate course of actions. Every course syllabus contains a clause relevant to cheating. For instance, at my university, the cheating policy contains the following sentences. "Cheating of any kind will not be tolerated. Any assignment or exam that is handed in must be your own work … If you let someone else copy your solution, you are cheating." Some universities even expel cheaters. A number of educators, including myself, disagree with such a policy. William Murray [8] writes: "I tend to see 'cheating' as a symptom that something is wrong in the system, in pedagogy." In my courses, I have never implemented the cheating policy in its strictest form, even when I caught a student cheating. I would talk to the students (the one who copied and the one whose assignment was copied) and let them know that the next time they violate the rules they will deal with the disciplinary committee. Typically, that has worked. I personally believe that it is not worthwhile to waste valuable professor's time trying to catch students cheating and then administering a punishment. In one of the first published papers on the topic[9], the authors are also of opinion that preventing cheating is much more preferable than trying to detect cheating and take action against the students involved. The Rutgers University School of Arts and Sciences includes a section on how to prevent cheating into their guidelines for new instructors [10]. In it, they recommend using "grading schemes designed to limit opportunities for cheating."

This paper presents the introductory computer science course cheating prevention strategy first introduced in [11]. It demonstrates how the described strategy essentially eliminates the problem of cheating in homework assignments. The paper first analyzes the problem of cheating. It examines the process of teaching and the conventional methods of evaluating students' understanding of the material. It demonstrates then how modifying the course syllabus helps the instructor to minimize students' cheating. The paper also describes the way of applying the proposed teaching approach. At the end, I present the results of the surveys he used in 2015-2018 to get feedback from the students who took his course.

2 Analysis of the Problem

2.1 What is Cheating and Why it is Bad

There have been published a considerable number of articles analyzing the problem of cheating. In [12], the authors define cheating, or plagiarism, in computer science assignments as "copying, modifying, or presenting another's source code. " The most popular cheating cases include the following [9]:

- Submitting someone else's slightly edited code as your own
- Allowing a fellow student to submit your homework as his/her own (in some cases, they even forget to change the programmer's name)
- Working as a team on an assignment that is supposed to be done individually

- Copying a part of someone else's exam
- Stealing an exam or solution from the instructor.

The majority of instructors notice that most of cheating takes place with homework assignments rather than exams [9]. Therefore, the last two cases of cheating are left out of the scope of this paper.

Some papers focus on the harm caused by cheating. The worst consequence of cheating is that by the end of the course a cheater neither understands the course material, nor possesses the programming skills he/she has been taught. Cheating is harmful because if even a few students cheat without being detected, the others feel demotivated [12]. Cheating is also bad because the students who copy their homework without even trying to understand the problem and the program used to solve it tend to put blame on the instructor in their course evaluations.

2.2 Applying the System Approach

In the article [13], I presented the teaching process as a system, including all its essential components. The instructor's main goal is to manage the class in such a way that the students learn and understand the course material. At the start of the course, they are provided with the course syllabus. Throughout the semester, the instructor gets feedback regarding how successfully his/her goals are being accomplished through the use of homework assignments, tests, or quizzes returned by the students. The instructor grades the homework, quizzes, and tests submitted by the students and derives the course grade (Fig. 1).

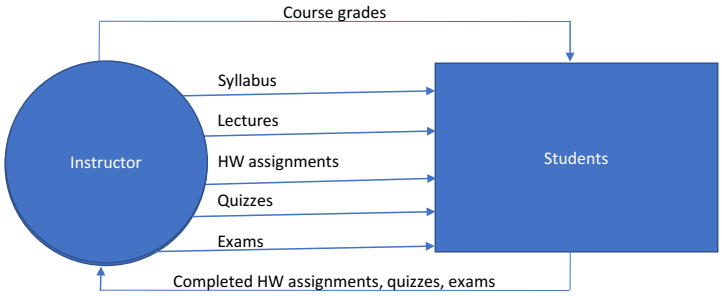

Figure 1: Systematic presentation of the teaching process

The instructor evaluates each student's performance based on his/her grades. It does not look logical if a student is doing great with the homework but poorly in the tests or quizzes, especially if the class assignments are open book. Although some students perform poorly under time pressure, generally the students getting low grades on the quizzes are the ones who copy homework assignments without trying to understand them. As a result, the students fail to demonstrate in their class assignments the knowledge and skills they were supposed to acquire in their homework assignments.

The instructor should help such students by explaining to them the material they have failed to understand. It works in many cases, but only if the students are willing to approach the instructor with questions. Unfortunately, very few students use that option. The more effective way for the instructor is to look critically at the course syllabus, the way the course material is presented, and the recommended textbook to determine possible changes/improvements in the way the course is taught. This paper concentrates on the course syllabus, particularly on the part of it that describes the course grading policy.

2.3 Analysis of the Course Syllabus

To be more specific, I would like to share my experience of teaching the computer programming course taken Freshmen not majoring in Computer Science. The course consists of two parts. In the first one, the instructor teaches how to use MATLAB to write simple scripts that include input/output statements, branching statements, loops, and functions. Students also learn how to write programs including file write and read operations. In the second part of the course, students learn some basic concepts of C++. For many years, the course grading policy was:

Homework (35%)

Quizzes (15%)

Midterm exam (25%)

Final exam (25%)

The homework assignments here were weighted so high because believed that the students would better learn computer programming if they were successful with the homework assignments over all the topics of the course. Usually, the assignments were fairly simple. Each of them emphasized the use of a particular programming concept. Students knew that if they submitted all or at least the majority of the homework assignments, a course-passing grade would be guaranteed. Therefore they tried to submit their assignments, even incorrect ones, on time (late submissions were not accepted). Such an approach led to cheating. A student whose goal was just to get a passing grade on the course did it easily. For instance, a student who earned 56 points out of 100 (which is an 'F') for his final exam could get over 75 points (a 'C+') overall. Such a low final exam grade clearly demonstrated the student's poor knowledge and understanding of the course material and was not consistent with the 'C+' final course grade.

3 Proposed Problem Solution

The straightforward method for solving the problem of cheating is to try to catch the cheaters, give them the 'F' course grade, and get the University Student Behavior Committee involved. The result is that the guilty are punished but the ultimate goal of the course - which is to have them learn computer-programming skills - is not achieved. Another approach is to "organize the course and computer usage to avoid situations in which students might be pressured or tempted to cheat" [9]. At the same time, the educational quality of the course should not be compromised.

The approach proposed here requires changing the course syllabus. First of all, the syllabus must clearly define what constitutes cheating. For example,

- Copying from a fellow student.
- Copying from the Web.
- Working together with a group of students on a non-group assignment .

I suggested to exclude the latter case from the syllabus. In other words, the situation in which a good student does a homework assignment and a poor student copies the assignment and takes credit for it is not considered cheating. But such a practice is justified only if the poor student actually understands the program he/she copied. The student who submits the assignment should also be required to report the name(s) of the student(s) he/she worked with. All students who are part of the group receive the same grade.

After one or more assignments on a given topic, a quiz on the topic is given to the class (each student is quizzed individually). The quiz reveals who of the students have actually understood their homework assignments. A similar method is described by [14]. Homework assignments are necessary because - by doing them - students develop new skills. The quizzes are important because they help the instructor evaluate students' basic understanding of the course topics. Quizzes therefore are more important and should have a larger weight than assignments. The above course grading policy should be changed to, for instance, the following:

Homework (20%)

Quizzes (35%)

Midterm exam (20%)

Final exam (25%)

Such a policy is in line with the Rutgers University cheating prevention guidelines that recommend instructors to "limit grades for collected homework to no more than even 10% of the total course grade, except in special courses where regular practice is deemed essential (C++ or other programming, final project work, etc) and which cannot be replaced by in-class quizzes or exams."[10] It is clear that with such a grading scheme excellent homework assignment grades cannot compensate for poor quiz grades. Each point of the quiz grade average is 1.75 times more valuable than a point of the homework assignment grade average. In other words, earning a homework grade average of 100 is equivalent to only around 57 points of quiz grade average. It means that the overall course grade will be poor if the quiz grades are low – assuming similarly low midterm and final exam grades. The probability of a student receiving high exam grades while getting low quiz marks is very low, since students who fail elementary quizzes will hardly succeed at the more challenging midterm and final exams. After each semester, the instructor should critically inspect the grades to determine if they reflect the students' knowledge and skills acquired in the course. If necessary, the grading scheme is adjusted.

An indirect proof of the described grading policy's effectiveness is that since it was first implemented, the author has never had to use grading on the curve.

4 Results of Applying the Proposed Approach

When the described teaching approach was first introduced, the author had the students answer a short questionnaire. The surveys were conducted at the end of the Fall 2015, Spring 2016, Fall 2016, and Spring 2017 semesters. The described approach was implemented in the Fall 2015, Spring 2016, and Spring 2017 semesters. In the Fall 2015, only one section of the course was taught and the approach was not fully implemented, as the students were not quizzed on all the course topics. But in the Spring of 2016, when two sections of the course were offered, the method was used fully on both. As it was expected, the correlation between the quiz grades and the final course grades was pretty high – 0.91 for the first section and 0.78 for the second. Also, the calculated average size of the homework assignment teams was 2.46 and 2.64, correspondingly. Interestingly, the first section's average course grade was 77.07, while the average course grade of the second section (with the larger average team size) was 81.48. Judging by the course evaluations, the method was well-received by the students. Useful information was also obtained by analyzing students' answers to a survey at the end of the Spring 2017 semester.

The results of the survey were published [13]. Surprisingly, shortly after that, in Fall 2018 and Spring 2019, the number of students interested in sharing homework assignments with their classmates was relatively small. I found a possible explanation to that phenomena in [15], who believes that students can be classified into two groups. One includes "freeloaders," the students who lack initiative

and avoid working hard. The second group are "lone wolves," the students who prefer working on their own, without collaborating with other students. Based on the definition given in [16], a lone wolf is "a psychological state in which one prefers to work alone when making decisions and setting/accomplishing priorities and goals." An assumption therefore can be made that the students who preferred not to share their homework with others where "lone wolves." Another possible explanation is the students' mental challenges, such as "anxiety, autism, or other issues that interfere with effective social interactions" [15]. I also believe that an important factor negatively affecting team formation was that the students who took the course were predominantly Freshmen and as such, were not acquainted with many of their classmates. Moreover, even finding a teammate did not guarantee that the cooperation would last. Some students worked together with their teammates only on selected assignments. The teams were not stable and often split.

5 "Selling" the Approach to Students

To encourage students to work on their homework assignments together with their classmates, I applied a strategy consisting of the following three steps.

- Help Them Find Partners
- Help Them Get Organized
- Teach Them Help Others

5.1 Help Them Find Partners

From the feedback received from the students who worked on their homework assignments individually, I learned that many of them did it because they were unfamiliar with their classmates. Another reason was that the students did not appreciate the value of teamwork. Lastly, the case of a student being a 100% "lone wolf" is always a possibility. Assuming the latter case to be an exception, the instructor from the very beginning of the course should help students to get to know their classmates. That can be effectively done by using the Discussion Board tool of Blackboard. The instructor should also explain to the students the importance of team work. By sharing with the students "success stories" about students' effective cooperation in the previous semesters when the course was offered, the instructor can motivate the class to at least "give it a try." The suggested strategy will be considered successful if 50% or more students find homework partners.

5.2 Help Them Get Organized

The last thing the course instructor needs is to have a small part of the students doing the homework assignments and letting their partners copy the results of their job. Such a practice is extremely de-motivating. A student with "lone wolfish" tendencies might decide that doing homework individually will involve less disturbance. He/she gets no benefit from letting their partner(s) copy their programs. The best-case scenario is where team members are equally involved in doing homework assignments. For instance, one team member writes a program, while the other does the program testing and debugging. It is also important that before the homework is turned in, it is unanimously approved by

both team members. Enforcing such a policy makes the team members feel equally responsible for the homework assignment and prevents "lone wolves" feeling like they are being "used."

5.3 Teach Them Help Others

The main purpose of homework assignments is having students better understand the course topics by applying their skills to specific world problems. Not all the team members are always capable of doing that, and there is nothing wrong if they ask their more advanced team members for help and/or explanation. The instructor should explain to students why they should not consider helping their partners understand a program being a waste of time. Contrary, when one student is explaining to the other a program code, he/she at the same time is testing the logic of the program. The efficiency of the process is further increased if the student being helped is not shy to ask questions. Such questions may lead to detection of program errors.

With an instructor utilizing the aforementioned three-step strategy, the teaching process can be schematically presented on Figure 2.

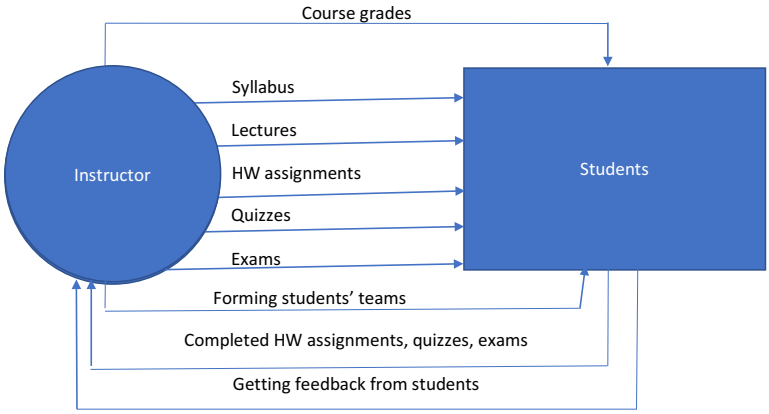

Figure 2: Modified presentation of the teaching process

In addition to the syllabus, lectures, homework, quizzes, and exams, the students receive help in forming teams with whom they will productively work for the whole semester. Receiving feedback from students helps the instructor to identify their needs and helps to improve the course.

6 Conclusions and Future Research

This paper presented an approach that can be used to practically eliminate Freshmen computer science students' cheating in their homework assignments. Instead of using the traditional methods of "fighting" cheating through administrative actions against the students found guilty, the proposed approach suggests changing the course syllabus and the way of teaching the course to permit students to work on their homework assignments together with others. As it is demonstrated in the article, homework assignments alone cannot reliably evaluate the skill level and understanding of a student. Instead, periodic quizzes are an important tool of verifying students' level of understanding of the

course material. The author used the proposed approach in the Fall 2015, Spring 2016, Fall 2016, and Spring 2017 semesters. The survey conducted among the students who took the course in Spring 2017 has demonstrated that the proposed approach:

a) Helps each student to understand every homework assignment and its solution.
b) Stimulates reading textbook and notes to better prepare for quizzes and exams.

The approach may also benefit the instructor by reducing the time spent on assignment grading. If student X reported he/she collaborated with students Y and Z, there is no need to run the programs submitted by Y and Z. On the other hand, the number of quizzes should be equal, at least, the number of topics covered in the course. The amount of time mentioned above may not be realized if, for instance, X reported he/she collaborated with students Y and Z but Z reported collaborating with only Y. In such a case, the instructor will have to clarify who actually collaborated with whom. Unfortunately, not all the students let the instructor know that they worked on the assignment with someone else. In such cases, the instructor's time is wasted on grading identical assignments.

Some instructors do not believe that weighing tests more than homework assignments is a good solution because "assignments test different things than exams" [4]. Their reason is that there are students who do much better on the assignments than they do on the written exams. Yet, many of them are good computer programmers. The proposed grading strategy does not contradict that opinion. The weights assigned to the midterm and final exams are the same as that of the homework assignments. The quizzes are much more valuable component of the course grade than homework assignments because they check students' understanding of the basics fundamental for program writing.
Working on homework assignments with other students is beneficial because it leads to better understanding of the course material and, ultimately, to better preparation to the quizzes and tests. The method works because it encourages collaborative learning, which has been found to be beneficial to many students [17].

Finding right partners and coordinating with them presents sometimes unfathomable problem for the students, which they cannot solve without the instructor's help. The instructor should explain to the students the value of teamwork, help them find partners, and get them organize. That can be achieved by using the three-step strategy presented in the paper. The first results of implementing the strategy have been encouraging. The class I am teaching now contains 26 freshmen. 22 of them (close to 85%) formed 10 teams of 2 or 3 students. The teams have been consistently working together on their homework assignments.

7 References

[1] Alcantara, C. (2012). University of Florida Students Caught Cheating on Computer Science Projects. *The Independent Florida Alligator*. March, 13, 2012.

[2] Bidgood, J., Jeremy B. Merrill. (2017). As Computer Coding Classes Swell, So Does Cheating. *The New York Times*. May, 29, 2017.

[3] Krieger, L.M. (2010). Stanford finds cheating — especially among computer science students — on the rise. *San Jose Mercury News, June 8, 2010.*

[4] Phillips, P., Cohen, L. (2014). Convictions of plagiarism in computer science courses on the rise. *The Daily Princetonian, March 4, 2014.*

[5] Wagner, N. (2000). Plagiarism by Student Programmers,

http://www.cs.utsa.edu/~wagner/pubs/plagiarism0.html

[6] Gabriel, T. (2010). Cheaters find an adversary in technology. *The New York Times, Dec. 27, 2010.*

[7] Silverman, M.P. (2015). Cheating or coincidence? Statistical method employing the principle of maximum entropy for judging whether a student has committed plagiarism. *Open Journal of Statistics, 5, 143-157.*

[8] Murray, W.H. (2010). Cheating in Computer Science. *Ubiquity, 2*.

[9] Shaw, M., Jones, A., Knueven, P., McDermott, J., Miller, P., Notkin, D. (1980). Cheating policy in a Computer Science department. *ACM SIGCSE Bull, 12(2), 72-76.*

[10] http://www.finmath.rutgers.edu/academics-finmath/for-new-instructors/149-advice-on-how-to-prevent-academic-dishonesty

[11] Sukhodolsky, J. (2015). Cheating Prevention in Computer Science Courses. *2015 Proceedings of the Information Systems Education Orlando, Florida USA, 371-378.*

[12] Chuda, D., Navrat, P., Kovacova, B., Humay, P. (2012). The issue of (software) plagiarism: a student view. *IEEE Transactions on Education, 55(1), 22-28.*

[13] Sukhodolsky, J. How to Reduce Cheating from an Introductory Computer Programming Course? *International Journal of Computer Science Education in Schools.* Oct 2017, Vol. 1, No. 4, pp. 25-34.

[14] Fraser, R. (2014). Collaboration, collusion and plagiarism in Computer Science. *Informatics* In Education, 13(2), 179-195.

[15] Finnegan, M. (2017, August 1). It's good till it's not. *Inside Higher Ed.* Retrieved from https://www.insidehighered.com/advice/2017/08/01/helping-diverse-learners-navigate-group-work-essay

[16] Dixon, A., Gassenheimer, J., Feldman Barr, T. Identifying the Lone Wolf: A Team Perspective. *Journal of Personal Selling & Sales Management*, vol. XXIII, no. 3 (summer 2003), pp. 205-2019.

[17] Chamillard, A.T., Kim A. Braun. (2000). Evaluating programming ability in an introductory Computer Science course. *SIGCSE '00 Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, 212-216.