DETECTING ACADEMIC MISCONDUCT IN INTRODUCTORY COMPUTER SCIENCE COURSES

Inés Friss de Kereki

Abstract: Most Computer Science programs start with introductory Computer Science courses ("CS1" and "CS2") where the fundamentals concepts of software development are introduced. In our courses, there are some large compulsory programming assignments to be done in pairs to develop the required basic skills. Since 2004, we are using different tools to detect plagiarism, that is, similarities between different students' works. Before 2004 we compared each work manually. Nowadays, due to their increasing number (cohort size CS1+CS2: +225 students), we use tools like MOSS which automatically detect similarities. We wanted not only to check the use of others code as their own, but also to check that both students of each team co-develop the programs. So, we included an evaluation ("defense") to check authorship and knowledge of the program. It is compulsory, must be done individually and consists of modifying specific details or correcting a mistake of their program. Also, we emphasized the importance of prevention, and developed some materials and recommendations for students to promote good and ethical practices in programming assignments. The detected number of plagiarism cases with automatic tools decreased from 3–6 each semester (2004–2012) to 0 in the last 4 years, possibly due to the prevention activities. With the "defense" we continue finding some cases of students not involved in the work they hand in. In this paper, we detail courses, strategies, a students' survey, and results.

Key words: Computer Science I, Computer Science II, Plagiarism, Programming

1 Introduction

Academic misconduct, according to Berkeley University, is "any action or attempted action that may result in creating an unfair academic advantage for oneself or an unfair academic advantage or disadvantage for any other member or members of the academic community."(Berkeley). Among others, they refer to cheating and plagiarism. "Cheating is defined as fraud, deceit, or dishonesty in an academic assignment, or using or attempting to use materials, or assisting others in using materials that are prohibited or inappropriate in the context of the academic assignment in question" (Berkeley). Plagiarism is "the practice of taking someone else's work or ideas and passing them off as one's own" (Oxford). IEEE (Institute of Electrical and Electronics Engineers) defines plagiarism as: "the reuse of someone else's prior processes, results, or words without explicitly acknowledging the original author and source." (IEEE). MIT (Massachusetts Institute of Technology) refers that "Cheating, plagiarism, unauthorized collaboration, deliberate interference with the integrity of the work of others, fabrication or falsification of data, and other forms of academic dishonesty are considered serious offenses" (MIT).

Some renowned universities have an "Honour Code". The Stanford Honour Code is "an undertaking of the students, individually and collectively: that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in

class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading" (Stanford University). Similar ideas are included in our "Honour Code" (Universidad ORT Uruguay). The University of Edinburgh includes in their Student Conduct Code examples of misconduct offences: "Assessment offences, including making use of unfair means in any University assessment or assisting a student to make use of such unfair means" (The University of Edinburgh).

MIT offers some concrete suggestions to students to avoid plagiarism: "Don't purchase papers or have someone write a paper for you. Undertake research honestly and credit others for their work. Don't copy ideas, data or exact wording without citing your source." (MIT 2). Also, it offers suggestions to avoid unauthorized collaboration: "Don't collaborate with another student beyond the extent specifically approved by the instructor.", and cheating: "Don't copy answers from another student; don't ask another student to do your work for you" (MIT 2).

Although clear recommendations about academic integrity are available for students like the previous ideas presented, academic misconduct is a big and prevalent issue in academia in general: "Plagiarism is a problem in probably the majority of high education institutions" (Hammond, 2002); "Plagiarism is a serious problem of today's copy-paste generation" (Sraka and Kaucic, 2009).

Computer Science (CS) is not an exception. In our case, almost in every course we detect some cases. This paper presents a unified proposal to try to mitigate that problem. The organization of this work is presented as follows. In Section 2 are concepts about academic misconduct. Also, some mechanisms to detect and mitigate it are included. Section 3 describes Computer Science I (CS1) and Computer Science II (CS2) in Universidad ORT Uruguay. Section 4 describes our proposal. In Section 5 the results are analyzed, and also include a students' survey. Finally, in Section 6 we present some conclusions and future lines of work.

2 Academic Misconduct

2.1 Academic Misconduct in Computer Science

CS1 is in the heart of computer science (Kinnunen and Malmi, 2006). CS requires a high level of abstraction at the cognitive level and a constant effort and dedication from the students (Silva-Maceda et al, 2016). Programming is a very useful skill but learning to program is difficult (Robins et al, 2003). Learning to program may be overwhelming for students (Price and Smith, 2014) and also to teach CS is complex (Vihavainen et al, 2011).

In CS courses, plagiarism and other forms of academic misconduct are not new. In 1989, Parker and Hamblen (1989) defined a plagiarized program as a program which has been produced from another program with small changes. In that year, they refer that student plagiarism of programming assignments has been made easier because of networks, easy to use editors, and large class sizes, among other factors. More recently, Sraka and Kaucic (2009) also refer to a higher number of students, while the number of staff is not increasing, so "the energy put into a single assessment will reduce". The problem is increasing (Jadalla and Elnagar, 2007) and serious: "Students have a practice

to copy and share their solutions rather than doing them by themselves" (Konecki et al, 2009).

"No matter how strict the system be, students somehow manage to copy" (Sharma et al, 2015). A student can copy the work of another student in the class or work together with other students to produce nearly identical code, or even work with students of different class (Koss and Ford, 2013). In particular, the situation when "two or more students submit either the same paper or very similar papers that have only had a number of superficial modifications done to them" is called "collusion" (Weber-Wulff, 2014). Collusion is an unpermitted group activity and "may be regarded as the middle ground in a spectrum of practices ranging from collaboration to outright plagiarism" (Fraser, 2014). Also, students can obtain solutions for the assignments from Internet (Sraka and Kaucic, 2009). Besides, they can hire an expert programmer for individual assignment (Koss and Ford, 2013). This is a case of "contract cheating": the submission of work for academic credit where the students had paid contractors to write for them (Clarke and Lancaster, 2006). Moreover, there are many sources like Facebook (Facebook) or StackOverflow (StackOverflow) – an online community for programmers to learn and share knowledge – where students can upload questions and obtain guides and code from other people. Sraka and Kaucic (2009) refer that some students used program assignments for previous generations. The social network formed in the classroom can act as a source of plagiarism (Luquini and Nizam, 2011).

To sum up, CS is a difficult area to learn, and plagiarism and other forms of academic misconduct are highly present nowadays in that area. There are a growing number of sources to obtain illegal help.

2.2 Detecting and mitigating academic misconduct in Computing Science

In CS, manual detection of plagiarism was found "to be inefficient but it is effort and time consuming", refers Jadalla and Elnagar (2007), so automated systems become essential. In those systems, the computer calculates the similarity rate of two coded files and generates a measuring standard; the user further decides whether the code is plagiarized with reference to this standard (Chen et al, 2011). MOSS (Measure of software similarity) (MOSS) is a free automatic system for determining the similarity of programs. It supports different programming languages like Java, C, and C++. JPlag (JPlag) is a system that finds similarities among multiple sets of source code files. Jadalla and Elnagar (2007) propose PDE4Java with similar results than JPlag. Sharma et al (2015) propose "Parikshak": "the results given by the tool endorse its effectiveness and usability". Z kov et al (2013) propose SIM, useful for detecting plagiarism in case of shorter program codes. Zhang et al (2014) note that those tools do not detect plagiarism originating from the Internet. Novak (2016) indicates that no similarity detection engine can be used to positively accuse someone of plagiarism: the teacher should always manually check the cases.

There are many proposals to mitigate academic misconduct. Roberts (2002) refers the approach taken in Stanford some years ago to control plagiarism in CS courses. It includes the use of electronic tools to detect instances of plagiarism, explicit departmental policies about collaboration and plagiarism, and make sure that those polices

are well understood by students (Roberts, 2002). Carroll and Appleton (2001) suggest changing assessments, creating individualised tasks, and integrating assessment task. This idea of frequent change of assignments is also referred by Z kov et al (2013). Similarly, Manoharan (2016) proposes personalized assessments with unique problem set for each student. Halak and El-Hajjar (2016) refer to unique assignments and individual presentation sessions, where each student is required to explain his-her own results to the class or to the professor; their evaluation indicates a reduction in plagiarism acts. As Sant (2015) mentions, these techniques (changing assignments, creating individualized assignments) are problematic for instructors of software development courses. Koss and Ford (2013) propose to check previous versions of the student's code. That would let instructors determine the student's development processes (Koss and Ford, 2013). Sant (2015) proposes repurposing code, to assess the ability to comprehend a code sample and use it to create a new solution.

Sraka and Kaucic (2009) suggest tackling the academic misconduct at different levels: using plagiarism detection systems, proper regulations, educating students about the topic, and proper assignments. Related to education, they refer to present the importance of authorship, intellectual rights, and rules of proper referencing (Sraka and Kaucic, 2009). Fraser (2014) suggests in CS coursework: "emphasizing the intended learning outcomes of each assignment, providing tutorial sessions to facilitate acceptable collaboration, delivering quizzes related to assignment content after each assignment is submitted", and "clarifying the boundary between collaboration and collusion in the context of each course." Pandey et al (2015) refer that "more seminars should be organized for students", where relevant case studies should be analysed and discussed. In their experience, those seminars contribute positively in student's perception and motivation against plagiarism practices (Pandey et al, 2015). Kumar and Abdus Sobhan (2006) summarise than over decades, some methods of detection and prevention have been proposed: a) the use of electronic tools to detect, b) improving learning and teaching ethics, c) penalizing persons for their guilt, and d) creating awareness about cyber law and copyright.

Also, we should consider student's own perspective about plagiarism. Zhang et al (2014) compare perceptions of Chinese students and UK students, and conclude that reuse source-code without acknowledging the original source and incorrect referencing appear to be issues in both countries. "UK and Chinese students are unsure of the boundary between legitimate academic activity and unacceptable plagiarism".

Dealing with plagiarism is time consuming: in detected cases, instructors need to conduct interviews, gather evidence, and fill administrative forms (Manoharan, 2016). Z kov et al (2013) suggest having a personal discussion with each involved student in a plagiarism case to see if he or she really understands the code and how it is working.

To sum up, academic misconduct in CS is an actual and open problem, with different proposals and tools to try to mitigate it, but, as Sharma et al refer: "cases of cheating have always been there as a part and parcel of examinations" (Sharma et al, 2015).

Table 1
CS1 description

Week	Topics
1–3	Variables, pseudo code, control structures
4	Classes and Objects
5–8	Relations between classes: Association, Inheritance
9–11	Relations between classes: Aggregation. Collections
12	Sorting and Searching
13–15	Advanced use of Collections.

3 Computer Science I and II Courses

3.1 Computer Science I

CS1 course is a first semester course at the School of Engineering of Universidad ORT Uruguay and is focused on teaching problem-solving methodology using Object-Oriented Programming (OOP). The course prepares the learner for constructing simple programs using that programming paradigm. By the end of the semester, the student will be ready to analyse simple situations, to design solutions and to implement them with an OOP approach, using Java as programming language. The duration of the course is 15 weeks, organized in 4 hours of lectures and 2 hours for lab session per week. There are 25–30 students in each class. Each year, we have two cohorts of CS1 students: March and August.

A brief description of the course is shown in Table 1. The main topics are: pseudo code, variables, and control structures, objects and classes, association, inheritance, aggregation and collections, sorting and searching, and advanced use of collections.

The course includes two relatively large programming assignments (done in pairs, 15 and 25 points each), in class participation (15 points) and a final written evaluation (45 points). Before 2016, there were no points for class participation, the assignments were worth 20 and 30 points and the written evaluation, 50 points. The first large assignment usually refers to develop some classes in Java to represent a situation in a business. The second assignment implies to expand that solution to include collections and inheritance. For those assignments each student is paired with another to work with, as Williams (1999) proposed as an useful application of collaborative programming to classrooms. The participation consists of 5 multiple choice quizzes, done individually in class and 4 short programming assignments to be solved by each student. These assignments are proposed early in the course to motivate and engage. To pass the course, the student must obtain 70 or more points. In case of less than 70 points, the student fails the course and must retake it.

3.2 Computer Science 2

This course in the second semester of the career continues the student's training in the paradigm of objects. It presents more advanced elements in the construction of

Table 2
CS2 description

Week	Topics
1–3	Data structures: arrays, maps. Algorithm design
4	Parameters. Aliases and references
5	Modelling systems: CRH (classes, responsibilities, helpers), use cases
6–8	User interface. Model-View-Controller
9	Pattern: observer
10–12	Streams. Persistence
13	Recursion
14–15	Exceptions

programs: reuse, polymorphism, modelling, and design. At the end of the course, the student can analyse more complicated domains, using methodology; design solutions and implement under the OOP paradigm. Like CS1, this course lasts 15 weeks: 4 hours of lectures and 2 hours for lab session per week. Java is the programming language.

The main topics are presented in Table 2. It includes: algorithm design, different data structures, tools for modelling systems, user interface design, persistence, recursion, and exceptions management.

The evaluation of the course is similar to CS1: two large programming assignments (also done in pairs, 20 and 25 points each), in class participation (10 points) and a final written evaluation (45 points). An example of the first large assignment may be to develop a game using bi-dimensional arrays. Interesting algorithms should be developed. Also, the design of the classes implies lots of though. The second assignment may include developing a Windows-like visual interface, persistence, modification of the model and more. Like in CS1, if the student obtains 70 or more points, the course is passed. In other case, it must be retaken.

4 Proposal

4.1 Background

Since 2004, we are using different tools to detect plagiarism, that is, similarities between different students' works. Before 2004 we compared the large programming assignments manually. This was feasible because the number of works were relatively small. For instance, in March 2003 we had 170 students in CS1+CS2 and in August 2003 we had 165 students in CS1+CS2. In March 2004 the numbers grown: there were 213 students in CS1+CS2. Due to their increasing number (nowadays in each semester there are approximately 225–250 students in CS1+CS2), we use tools like MOSS (MOSS) and JPlag (JPlag). Since 2004, we keep records of the detected academic misconducts and the reasons argued by students, after the corresponding analysis. Most representative cases are presented in Table 3. The specific circumstances were considered when deciding penalties by the Director of the area and those penalties vary from loss of some

Table 3
Most representative cases detected (2004–2012)

Year	**Case**
2004	Using partial work of other group as owns code
	Different groups present identical solutions
2005	Using code of other student without authorization
2006	Task developed by more than 2 students
2007	Use of previous year work of other student as owns code
	Giving the work of one student to other, without knowledge of the original author
	Receiving too much help from other group
2008	Using code obtained without authorization from another computer in lab
2009	Two works almost identical
2010	Using code from other student, obtained by not authorized access in the lab network
2011, 2012	More than 2 students developed a task

points to suspension of academic rights during a year. In the last 4 years (2013–2016), no cases of plagiarism have been detected neither with MOSS nor JPlag in CS1 and CS2. A similar situation is reported by Kraemer (2008): plagiarism decreased in their courses since he began using a plagiarism catcher computer program.

4.2 Our proposal

Currently, our proposal against academic misconduct in CS1 and CS2 combines aspects referred by Kumar and Abdus Sobhan (2006) and Z kov et al (2013), among others, but also including personalized instances called "defenses": we wanted not only to check the authorship of the code, but also to check that both students of each team co-develop the programs, to detect situations like using code of others, obtaining external assistance in the development, buying code to a professional programmer, and so on. We promoted lab based examinations, as suggested Culwin et al (2001) and Barros et al (2003).

As shown in Fig. 1, to avoid misconduct the complete proposal includes: a) prevention, b) use of automatic tools, and c) "defenses". In case of detecting any problems, we: d) penalize, and e) keep records of the problems, which will be included in future prevention, as feedback.

Related to prevention:

- We created a "Decalogue of good practices" and it is discussed in class. It includes recommendations like: "Start the task as soon as possible", "Be selfish: the code is personal, do not share it", "Backup your work", "Prepare the final version with enough time", and "Check the final version in the labs".

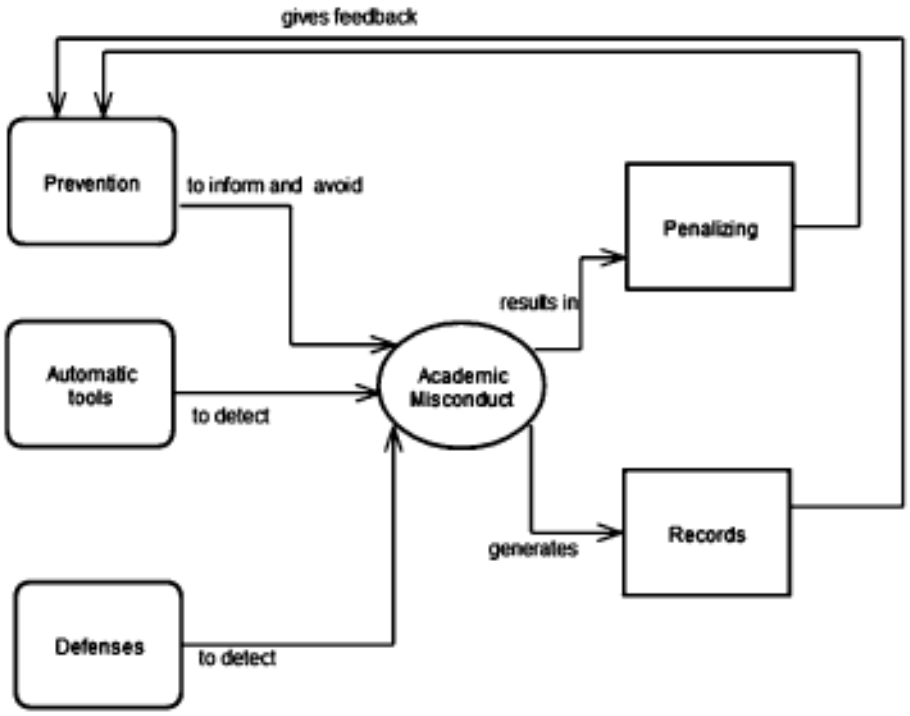

Figure 1. Proposal against Academic Misconduct in CS1 and CS2

- We showed and discussed with students a presentation of highlights of MOSS, and how the results are presented. We emphasized that "we check all the programs, it is not a myth, it is real!"
- We clearly informed the students the University' policies against plagiarism and penalties.
- We created different tasks for each course, to avoid sharing solutions among previous year students. The idea of unique assignment specification for each student as propose Halak and El-Hajjar (2016) may not be applicable due to the size of our cohorts.
- We removed the forums of the courses. The forums in the website of the University are not-moderated. This allowed that some students uploaded solutions of tasks without any control. Although could be argued that closing the forums was a questionable idea because it may push students to private forums or other social networks to share code, we did not want to endorse or facilitate that practice inside the University. We encouraged personal questions via e-mail or in person (there are 10 hours per week of teaching assistants).

- We promoted an early engagement with the course with short and easy tasks at the very beginning of the course. For example, in CS1 we ask the students to upload a Scratch program (Scratch).
- We promoted the knowledge of the "Honour Code" of our University (in class we mentioned it and its link in the website of the University to further references).

Related to automatic tools:

- We used MOSS (MOSS) systematically for the large Java assignments. In some semesters, we used also JPlag (JPlag). For each assignment, we upload all students' tasks and compare. Each class is labelled with the initials of the teacher (for instance "AD"), and each group is numbered consecutively ("AD01", "AD02", "AD03",...). To simplify this, our staff developed a Java program which uses the teacher' initials, students' directories and Java files as input (with one folder for each pair of students), and creates the respective folders for MOSS. We compare them all together and evaluate the results. In a similar way as Novak (2016) expresses, the pairs with higher similarity results are checked and the teacher should take into account that code, for instance, if that code are "setters" and "getters" and not relevant for the core assignment. In case of detection of any problems, we informed to the Director of the School and she took the corresponding actions.

Related to "defense":

- We included individual and personalized "defenses". Some years ago, we used an oral examination – viva – in pairs (both students who worked together in the task) as "defense". The teacher asked oral questions to each pair of students about certain aspects of the task. Frequently, the student who took the leading role in the development of the task tried to answer all the questions, relegating the other member. We modified the "defense" to specifically ask each member some questions. In very few cases it was detected students who had not worked in the task, and it was difficult to document the situation for which it would later be penalized. So, we changed the procedure to "individual defenses" which involves changes in student's work.
  For example, if the task in CS1 asked "to list customers who bought more than a certain amount", we may request: "to list customers who bought between X and Y", it is a similar algorithm. It implies knowing how the customers were stored; going through the necessary lists and filtering, with a somewhat broader criterion than they already did at work. In CS2, if the task includes the use of a visual interface with a grid for a game, we could ask to paint some cells (borders, diagonal, etc.). It implies to know how the data is represented, how is presented and to develop an algorithm.
  If the task was previously corrected by the teacher, the changes could refer to a particular problem detected in the student's solution. An example could be: "in your task, if we remove the last two elements of the list of clients, the program crashes. Correct that problem". Weber-Wulff (2014) suggests "to be attuned to unusual language" to detect "contract cheating". In programming it may refer to unusual solution's design, particular algorithms, or non-common programming

style. In those cases, the change may be specifically referred to any of those elements.
At the beginning of the "defense", the time is recorded and the required changed is presented. The description is written down. The proposed change is the same for both members. The original program is delivered to the students, each in one computer. It is student's responsibility to load it into the computer and to edit it. Then, each student must program the change and when he or she estimates it is ready, ask the teacher to verify it. Students have one hour and a tolerance of 10 minutes for the development. If he or she makes the change successfully, the "defense" is approved, so he or she receives the original qualification of the work. If the change is not correct, and, or, the time is exceeded, it is recorded in the sheet details of this situation, leaving clearly documented the case. If is it needed, the modified code is saved too, for further revision. Both the teacher and the student sign the sheet. These situations involve loss of points (possibly all of the points of the task) and are analysed together by teachers, to unify criteria. With the "defense", we usually detect 2–3 students per class (of 25–30 students) who cannot program the required modification.

If any case of misconduct is detected, the situation is recorded and used as input for prevention. The teachers directly do the penalization, in case of failed "defenses", or by the Director of the School in case of plagiarism or other forms of academic misconduct. According to the particular case, the penalization could be from loss of some points of the task (in the case of excessive time to do the change in the program), fail the course and, or, a suspension of student's rights for some time.

5 Results

With automatic tools, the detected number of plagiarism cases decreased from 3–6 each semester (2004–2012) to 0 in the last 4 years, possibly due to the prevention activities. With the "defense", we continue detecting cases (2–3 in each class of 25–30 students). None of the curriculum of both courses was changed. The final written exam and large programming assignments maintained the same structure.

The previous format with oral instances were not as useful due to personal characteristics of the students and, or, the difficulties to record the exact situation. Also, it is time consuming. For us, the current format of the "defense" (as we presented, implementing a particular change in the code), is more effective. In our labs 20–25 students can work simultaneously. After each student completes the change, it takes few minutes to the teacher to check it. We can check about 6–8 groups in an hour. We include all the "defenses" of each class in the respective 2 lab hours of the week. In each course, there are two instances of "defense", one for each large assignment.

In December 2016, we conducted an anonymous survey to ask CS2 students about their perceptions about academic misconduct and our proposal. To clarify, we included at the very top of the survey a detailed explication about academic misconduct and particularly plagiarism. We obtained 115 answers (of 122 participants in CS2). The results are summarized in Table 4.

Table 4
Academic misconduct: Students' opinions

Question	**Answers (115 answers)**
Do you know anybody who committed plagiarism?	Yes: 13.04% (15 students) No: 86.96% (100 students)
Did you commit plagiarism in CS1/CS2?	Yes: 5.22% (6 students), none of them was detected No: 94.78% (109 students)
Did your teacher emphasized against plagiarism and promoted good practices?	Yes, a lot: 49.57% (57 students) Yes, some: 40% (46 students) No: 10.43% (12 students)
In your opinion, committing plagiarism is...	Not ethical: 84.35% (97 students) Indifferent: 8.69% (10 students) No opinion, do not answer: 6.96% (8 students)
Do you know the "Honour code"?	Yes, I read it: 20.87% (24 students) Yes, but I did not read it: 40% (46 students) No: 39.13% (45 students)
Do you know about the use of antiplagiarism software in the course?	Yes: 98.26% (113 students) No: 1.74% (2 students)
"The defense is useful to validate authorship of the tasks". Your opinion is:	Yes: 65.22% (75 students) No: 24.35% (28 students) I do not know: 10.43% (12 students)
In your opinion, main reasons to commit plagiarism are:	Time problems: 58.26% (referred in 67 responses) Task difficulty: 55.65% (64 responses) Task overloading with other subjects: 50.43% (58 responses) Missing motivation: 44.35% (51 responses) Team problems: 13.91% (16 responses)

13.04% of students referred that they know somebody who committed plagiarism, and particularly 6 students (5.22%) indicated that they committed it, none of them were detected. Sraka and Kaucic (2009) refer the influence of the professors on plagiarism. In their survey, 53.4% of students (78 students) strongly agree with the statement that the professors "clearly stated that plagiarism is not desirable". In our case, almost 90% of the students (103 students) expressed that their teacher emphasized against misconduct.

Almost all students consider "not ethical" to commit plagiarism (84.35%). 60.87% of the students refer knowing the existence of an "Honour Code", but only 20.87% read it. Students know the use of antiplagiarism software: almost all of them indicated that they know that practice. Related to the "defense", 65.22% of the students refer that is useful to validate authorship, 10.43% indicated that they "do not know" and 24.35% refer that is "not useful".

The main reasons to commit plagiarism cited by students are: time problems, task difficulty, and task overloading with other subjects. We can relate those topics to time management and organizational skills. Those results are in the same line, according to the results of the survey of Pandey et al (2015): 80.2% of their participants refer that poor time management and lack of organizational skills are a cause of plagiarism.

6 Conclusions and future work

In this paper, we present our integral proposal against academic misconduct in CS1 and CS2. It includes: prevention, use of automatic tools, personalized "defenses" (instances of coding personalized changes of code in a limited time), and, in case of problems, penalizations and keeping records of the situation, to use that information as feedback.

When we used only automatic tools, some cases were detected, but that number decreases as students noticed that those tools are certainly used and powerful. With the introduction of the "defense", the number of detected cases is 2–3 of each class of 25–30 students. Academic misconduct is still present based on the results of anonymous students' survey (5.22% expressed committed undetected plagiarism) and we must continue working on that topic.

As future work, we propose personalized interviews to try to understand why some students (24.35%) indicated the "defense" as "not useful" to validate authorship and to discover missing aspects to avoid misconducts. Moreover, we plan to introduce more activities in CS1 to promote the detailed knowledge of the "Honour Code". One of these may be a Scratch (Scratch) animation including it. Also, we plan to include activities in CS2 to promote reflection about misconduct. It may be useful adapt the case-scenarios of plagiarism to discuss in groups, for instance: "Mary found in Internet an algorithm that solves a requirement of a task. She uses it directly and puts in the code the comment that the algorithm was found in Internet". Is this a case of plagiarism or not?

Literature

Barros, J., Estevens, L., Dias, R., Pais, R., Soeiro, E. (2003). Using lab exams to ensure programming practice in an introductory programming course, *ITICSE 2003, Greece*

Berkeley University: Definitions & Examples of Academic Misconduct. Accessed Jan 2017, http://sa.berkeley.edu/conduct/integrity/definition

Carroll, J., Appleton, J. (2001). Plagiarism: A Good Practice Guide. JISC Report, 2001

Chen, G., Zhang, Y., Wang, X. (2011). Analysis on identification technologies of program code similarity. Proc 2011 Int. Conf. of Information Technology, Comp. Engineering and Management Sciences

Clarke, R., Lancaster, T. (2006). Eliminating the successor to plagiarism? Identifying the usage of contract cheating sites. In: Second International Plagiarism Conference: Prevention, Practice and Policy, Newcastle, UK

Culwin, F., MacLeod, A., Lancaster, T. (2001). Source code plagiarism in UK HE Computing Schools. 2nd Annual LTSN-ICS Conference, London.

Facebook: Accessed Jan 2017, https://www.facebook.com

Fraser, R. (2014). Collaboration, Collusion and Plagiarism in Computer Science Coursework. Informatics in Education, Vol. 13, No. 2, 179–195

Halak, B., El-Hajjar, M. (2016). Plagiarism detection and prevention techniques in Engineering Education. 2016 European Workshop of Microelectronics Education.

Hammond, M. (2002). Cyber-Plagiarism: are FE Students getting away with words? Association of Northern Ireland Colleges

IEEE: Introduction to the Guidelines for Handling Plagiarism Complaints. Accessed Dec 2016, https://www.ieee.org/publications_standards/publications/rights/plagiarism.html

Jadalla, A., Elnagar, A. (2007). PDE4Java: Plagiarism detection engine for Java source code: a clustering approach. Proc of iiWAS2007

JPlag: Accessed Dec 2016, jplag.ipd.kit.edu

Kinnunen P., Malmi, L. (2006). Why students drop out CS1 course?. Procs. of the Second International Workshop on Computing education research (ICER '06), UK.

Konecki, M., Orehovacki, T., Lovrencie, A. (2009). Detecting computer code plagiarism in higher education. Procs of ITI 2009, Croatia

Koss, I., Ford, R. (2013). Authorship is continuous: managing code plagiarism. IEEE Security & Privacy, vol. 11, pp. 72–74, March–April 2013, doi:10.1109/MSP.2013.26

Kraemer, D. (2008). Using a Plagiarism-Catching Computer Program as a Teaching Tool, Proceedings of the 2008 ASEE North Midwest Sectional Conference

Kumar Dey, S., Abdus Sobhan, M. (2006). Impact of unethical practices of plagiarism on Learning, Teaching and Research in Higher Education: some combating strategies. ITHET 2006

Luquini, E., Omar, N. (2011). Programming Plagiarism as a Social Phenomenon. 2011 IEEE Global Engineering Education Conference, Jordan

Manoharan, S. (2016). Personalized assessment as a means to mitigate plagiarism. IEEE Transactions on Education, Issue 99

MIT: Policies and Procedures. Accessed Jan 2017. http://web.mit.edu/policies/10/10.2.html

MIT (2). Academic Integrity at MIT. Accessed Jan 2017, https://integrity.mit.edu/handbook/academic-integrity-mit/what-academic-integrity

MOSS: Accessed Dec 2016, https://theory.stanford.edu/ aiken/MOSS/

Novak, M. (2016). Review of source-code plagiarism detection in academia. MIPRO 2016, Croatia

Oxford Dictionary, Accessed Jan 2017, https://en.oxforddictionaries.com/definition/plagiarism

Pandey, A., Kaur, M., Goyal, P. (2015). The menace of plagiarism. 4th Int. Symp. on Emerging Trends and Technologies in Libraries and Information Services

Parker, A., Hamblen, J. (1989). Computer Algorithms for plagiarism detection. IEEE Transactions on Education, Volume 32, No. 2, May 1989

Price, K., Smith, S. (2014). Improving student performance in CS1. Journal of Computing Sciences in Colleges, Volume 30 Issue 2, pp 157–163, Dec 2014

Roberts, E. (2002). Strategies for promoting academic integrity in CS Courses. 32th FIE, Boston, USA

Robins A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, Volume 13, No. 2, pp 137–172.

Sant, J. (2015). Code Repurposing as an Assessment Tool. ICSE 2015, Italy

Scratch: Accessed Dec 2016, scratch.mit.edu

Sharma, S., Shekhar Sharma, C., Tyagi, V. (2015). Plagiarism detection tool "Parikshak". 2015 Int. Conf on Communication, Information & Computing technology, Mumbai, India

Silva-Maceda, G., Arjona-Villicaa, D., Castillo-Barrera, F. (2016). More time or better tools? A large-scale retrospective comparison of pedagogical approaches to teach programming. IEEE Transactions on Education, 59(4), pp 274–281

Sraka, D., Kaucic, B. (2009). Source code plagiarism. Proc. Of ITI 2009 (31st International Conf. on Information Technology Interfaces), Croatia. pp 461–466

StackOverflow: Accessed Jan 2017, http://stackoverflow.com/

Stanford University: Stanford Honour Code. Accessed Dic 2016, https://communitystandards.stanford.edu/student-conduct-process/honor-code-and-fundamental-standard

The University of Edinburgh: Code of Student Conduct. Accessed Jan 2017, http://www.docs.sasg.ed.ac.uk/AcademicServices/Discipline/StudentCodeofConduct.pdf

Universidad ORT Uruguay: Honour Code. Accessed Dec 2016, http://www.ort.edu.uy/varios/pdf/codigodehonor.pdf

Vihavainen, A., Paksula, M., Luukkainen, M. (2011). Extreme apprenticeship method in teaching programming for beginners. Proc. of SIGCSE 2011, USA

Weber-Wulff, D. (2014). False Feathers. Springer Berlin, Germany.

Williams, L. (1999). But, Isn't that cheating?. Proc. of 29th Frontiers in Education Conference, Puerto Rico.

Záková, F., Pistej, J., Bisták, P. (2013). Online tool for student's source code plagiarism detection. ICETA 2013, Slovakia

Zhang, D., Joy, M., Cosma, G., Boyatt, R., Sinclair, J., Yau, J. (2014). Source-code plagiarism in universities: a comparative study of student perspectives in China and the UK. Assessment & Evaluation in Higher Education, Vol 39, No. 6, pp. 743–758

Copyright statement

Author

Inés Friss de Kereki (kereki_i@ort.edu.uy), Engineering School, Universidad ORT Uruguay, Cuareim 1451, 11100, Montevideo, Uruguay