Decoding code plagiarism

Vreda Pieterse
Department of Computer Science
University of Pretoria
Pretoria, South Africa
vreda.pieterse@up.ac.za

ABSTRACT
The topic of plagiarism in academia is as old as academia itself. It has, however, received renewed interest as a result of the recent technological advances that make it easier to cheat. This situation has also sparked the development of sophisticated tools that can be applied to detect cheating. Much has been written to clarify what constitutes plagiarism in academia and what should be done to sustain integrity in academic writing. The same cannot be said about plagiarism with reference to writing code. This paper presents arguments to illustrate why it is much harder to define code plagiarism and consequently very difficult to curtail in an introductory programming module.

Categories and Subject Descriptors
K3.2 [Computers and Education]: Computer and Information Science Education—computer science education

General Terms
Human factors

Keywords
Cheating, code plagiarism, undergraduate students

1. INTRODUCTION

The notion that the moral and ethical standards of the world population are diminishing is widely shared. In academia this notion is supported by a perceived increase in the prevalence of cheating and plagiarism. Lecturers have reason to be anxious because they are not able to control how students increasingly use freely available electronic information.

The problem of plagiarism and academic dishonesty is widely discussed. Blum [6] suggests that this problem arises primarily from a lack of communication between two distinct cultures within the university setting. On one hand, professors and administrators regard plagiarism as a serious academic crime, an ethical transgression, even a sin against an ethos of individualism and originality. Students, on the other hand, revel in sharing, in multi-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

Copyright is held by the owner/author(s).
SACLA 2014, 25-26 June, Port Elizabeth, South Africa
other seems to rise. Suleman [22] reports that his teaching assistants voiced their concern about increased probability of copying that may occur when the code is not scrutinised by people during manual assessment. Chen [7] remarks that cheating is always an issue, especially where projects are reused. Ihantola et al. [10] maintain that students might be more inclined to knowingly submit weak or incorrect solutions that get accepted by a machine than trying to cheat a person. For this reason automatic assessment systems often incorporate plagiarism detection; for example, plagiarism detection is an integral part of BOSS [11]. Many dedicated systems for program similarity testing are available, of which MOSS [2], JPlag [18], Plagie [1], YAP3 [25], SID [8], SSID [17], and PlaGate [9] seem to be relevant in educational settings.

We discuss three ways in which we gathered information about how well the students in our module understand what plagiarism is and to what extent they are involved in plagiarism. In each case we describe how these incidents are related to our total teaching delivery to educate our students about this matter. In Section 2 we elaborate on how we involved students in an on-line discussion on the topic. This exercise revealed some misconceptions that could be cleared and provided an opportunity to offer arguments that may have improved the students’ insight. In Section 3 we provide data about the submission and assessment of the students’ solutions to practical assignments and how the decision to follow up on observed cases of plagiarism has changed student behaviour. In Section 4 we describe the answers students have given in a written test. It shows that the majority of students currently registered for this module are well informed about plagiarism and aware of its consequences.

2. ON-LINE DISCUSSION

As part of the presentation of our module, we provide an on-line discussion forum. The use of on-line forums in teaching is widely accepted and used to address the problem of diminished contact with instructors and peers. Jungic et al. [12] suggest that discussion forums, among other things, can contribute to increased interaction with students. Pieterse and van Rooyen [16] concur. Participation in on-line forums may have a positive effect on student learning [5].

In our case, the course discussion forum was employed to provide an opportunity for students to delve deeper into the meaning and consequences of plagiarism and how it relates to code reuse. A discussion type question on the topic of code reuse and plagiarism was opened with the following introduction:

> There is a fine line between the concepts of code reuse and plagiarism. The concept of plagiarism by itself is problematic owing to unclear definitions and unclear rules. See for example http://en.wikipedia.org/wiki/Plagiarism. An aspect that adds confusion – when referring to code – is the idea of code reuse. Code reuse is commonly advocated to save time and energy. Students are invited to share their ideas about these topics in order to arrive at a satisfactory definition of code plagiarism in the context of an academic setting.

The thread was intended to encourage students to think about the definition of plagiarism and its consequences in the context of writing code in our course. Students were expected to contrast plagiarism of code with the principle of code reuse. The thread had 83 posts involving 64 of a total of 431 students (14.85%) who were registered for the module at the time of this discussion.

Although the number of students who participated was within expectation, the quality of their participation was disappointing. Much of the disappointment is related to students voicing their opinions about these concepts without proper knowledge of them. More than 70% of the posts did not add value.

These posts were classified in categories. Figure 1 shows the number of posts that were classified in each of these categories. Note that the total does not add to the number of posts because some posts were classified in more than one category.

Some of the interesting conversations that were observed on the forum are discussed in the following sections.

2.1 Definitions

Many posts described code reuse or plagiarism accurately. The following textbook definition appeared as the starting sentence of the fourth post. It also appeared in several subsequent posts:

> Once you use a code that is not yours it is plagiarism in my own opinion.

Most of the posts that offered definitions also provided examples or counter-examples to illustrate these concepts. The following post by one of the students clearly defines plagiarism, brings forward the difficulty in knowing if code was plagiarised, and concludes with a moral remark:

> If you copy someone else’s code and claim it as your own is undeniably plagiarism. However; my opinion is that there aren’t really many things you can do differently when writing a code. It’s undeniable that some codes will look almost identical. Calling that plagiarism just isn’t correct. Reusing; to me; is using the basic blocks of programming to write a program. If you’re unable to write even a simple code without help and require assistance; it clearly shows you have not taken responsibility for your own work.

2.2 Scenario queries

Each post that was classified as a scenario query describes a specific scenario and then asks if this would mean that the code involved was plagiarised. Many of these posts displayed ignorance. Some were answered but most remained unanswered.
This illustrates that there are many grey areas where it is difficult to decide if a specific action is an act of plagiarism or not.

A prominent issue is the use of code that appears as an example in the textbook. Another issue that received a lot of attention is the extent of help from tutors and peers that is acceptable. Students are also unsure how they should reference their sources in code. The following are examples of posts that remained unanswered:

*let's say you are using an example from the textbook to complete your program; and you know nothing about codes like I do because you were not exposed to programming in high school; is that also plagiarism?*

*My question is: if you are struggling with a piece of code and you work together with a fellow student to resolve the problem; where both contribute; (which would result in very similar code) would it still be considered plagiarism?*

### 2.3 Misconceptions

A standard misconception of the definition of by code reuse started to emerge. Many students seem to be under the impression that when one alters existing code to suit a specific need that is very similar to the intention of the original code, then one is reusing the code. The following post defines plagiarism accurately but is a classic example of the erroneous viewpoint that hacking is a form of code reuse.

Plagiarism basically means copying information which you do not truly understand creating an impression that you truly understand whatever it is you are engaged in. Therefore why you engage in code reuse answers the question whether you are engaged in plagiarism or not.

Some posts indicated that many students are ignorant about plagiarism. Some students seem to be under the impression that as long as the work that is submitted cannot be clearly identified as plagiarism, then it is acceptable. The following post illustrates that this student was convinced that the hacking of code in this manner does not constitute plagiarism:

*Once you use a code that is not yours it is plagiarism in my own opinion. Code re-use may be presented only by the use of same pattern of processing of your code. The second code should be consist of different functions from the other (e.g. use of different data types; etc) in a different style. Other than that it is plagiarism.*

Some even offered situations in which it might be justified to copy or hack code with the wrong intentions. The following post clearly purports that it is justified to copy code if one is under the impression that one would have been able to write the code oneself if only one had more time:

*It is unfair to label code reuse as plagiarism when it is done for the good reasons. If the individual reusing the code knows how to create a similar code but is honestly doing so to save time then I don’t feel that’s plagiarism.*

The following excerpt insinuates that it is acceptable for novices to hack code as a means of learning:

Somsies; code generation requires extensive computer-programming knowledge; which is not possible for novice programmers; as they are still in the learning phase.

### 2.4 Insight and good advice

Fortunately the discussion led to the realisation of misconception and ignorance about what code reuse entails. The following is an example of a post illustrating this:

*I think what you describe here is closer to plagiarism than to code reuse. So far I have not seen a good enough explanation of what code reuse really means and why code reuse would not be plagiarism.*

It is encouraging to observe that some students have the right attitude and will put in the effort to understand concepts and write their own code after grasping the ideas needed to create a solution to a given task. The following post gave this advice to fellow students:

*I believe if you find yourself in a situation where you can’t go forward with a programming task; its best to find someone to help you by understanding the concepts behind the question requirements and the simplest logic to finding solutions to these requirements. This will allow you to become a full-fledged programmer that thinks before they code.*

It is unethical to fool a person or a system to create the impression that you understand something or are able to do something. The following post gives advice on how one can honestly learn by using existing code and also points out that dishonesty is futile.

*Study the code and understand what it does. Then code your own solution; this makes you better at coding and eliminates the need to plagiarise. If you just copy code all the time what do you learn?*

### 3. Submission of Copied Solutions

#### 3.1 Context

We have been using a system for the automatic assessment of programming assignments for several years [15]. Prior to our intervention to address the submission of copied work, we suspected that some of our students would not hesitate to upload someone else’s code. We endeavored to determine the extent of the occurrence of such malpractice. We gathered data about the number of students who could be identified to have submitted copied work beyond doubt relative to the number of students who submitted work.

The number of assignments that are given to students differ from year to year. Furthermore, the time at which these assignments are given, as well as the difficulty of the assignments, vary largely. In this investigation we report only on three comparable assignments. For each year within our investigation period, we selected an early assignment, an assignment that was given in the middle of the semester, and one towards the end of the semester. These were selected to portray similar scope and difficulty levels.
This phenomenon can be attributed to general tendencies. The figures for 2014 for the middle and late assignments are not yet available. In 2012 and in 2013 the number of uploads dropped drastically after the first few weeks, but after the drop, remained more or less constant towards the end of the semester. This can be attributed to two factors. Firstly, the number of registered students usually decreases early in the semester owing to de-registration. Secondly, the students tend to lower their participation levels in all course activities once they are familiar with the course presentation style. Students who are active one month into the module seem to stay active for the rest of the semester.

3.2 Identification of copies

To identify copied work, we applied comparison of MD5 hashes of the uploaded archives. MD5 is believed to produce a unique 28-bit value for any given file. It is commonly used to verify data integrity [24]. If two archives produce the same MD5 hash, it can be assumed that the content of all the files in those archives are identical in every aspect. We realise that this can only identify exact copies. If a student altered the copied code in any way, the MD5 hash of the copied code would differ from that of the original code. The addition or removal of a mere space character will render a different MD5 hash, let alone more elaborate changes that can be applied to avoid detection. We know that this technique identifies only blatant exact copies while it is possible that many plagiarised submissions may remain undetected.

A script was written to calculate the MD5 hash for each archive that was uploaded. When more than one upload with the same MD5 hash value occurred, the occurrences were gathered and the identity of the owners of these archives was established. Where more than one occurrence of the same file were submitted by different students, students who were involved in these uploads were listed and then counted. This procedure was followed on all the uploads in the chosen assignments. Figure 3 shows the percentage of students who were identified to have submitted exact copies for these assignments. Data for 2012 shows a sharp drop between the beginning and the middle, but an alarming increase towards the end whereas the 2013 data shows a steady decline. We expect that the 2014 data will show a pattern similar to the 2013 data. This phenomenon can be attributed to general tendencies as well as to our intervention that is discussed in the next section.

3.3 Intervention

During 2012 no effort was made to follow up on the culprits, while in 2013 the students who were identified to have participated in this practice were called in. The students were granted an opportunity to give their justification for uploading the same solution. They were also lectured on the consequences of copying solutions rather than programming their own. The students who were involved in this anti-plagiarism training included those who used solutions of others as well as those who supplied the solutions.

When looking at all assignments of 2012 to 2014 so far, higher occurrences of copying coincided with the tasks where the average marks of the students were lower. This confirms the observation by Ramzan et al. [19] that difficulty in the assignment is one of the top reasons for plagiarism. The decline in observed cases of copying between the early assignment and the assignment in the middle of the semester in both 2012 and 2013 may be because the students who participated generally found this assignment easier. The students who found the first assignment difficult and had resorted to copying then, had probably dropped out at this stage.

The sharp contrast in copying behaviour between the 2012 and 2013 groups for the later assignment can be explained by combining Ramzan et al.’s observation and the impact of our intervention. In 2012 those who remained in the module and still found the tasks manageable in the middle of the semester, may have found the tasks increasingly more difficult towards the end of the semester and consequently more frequently resorted to copying. They were not as concerned about the consequences of plagiarism as their 2013 counterparts.

The intervention involved only a small selected group of students. We believe that communication between students allows the message to spread effectively. This can be seen in the lower occurrence of copies identified in 2013 when compared to 2012. A decline can be observed in the early assignment of 2013. This particular assignment was the second assignment of 2013. A small number of students had already been called in about transgressions that were observed during the first assignment of 2013. The transition of this community learning over years can be observed when comparing the copying behaviour of the students in the early assignment of 2014 with that of the previous years. It is heartwarming to see evidence that our intervention has a positive effect. We realise that this merely shows that the students know how to avoid detection of blatant copies and cannot be interpreted as a reduction of copying and hacking in general.
3.4 Student justification

Students who claim that they have worked together and created a single solution collaboratively were surprised that staff identified the identical solutions they submitted as plagiarised. Many students deemed it justifiable to upload someone else’s correct solution when their own seemed to be inadequate. A student who was identified to have participated in this practice said:

My program was exactly the same as his. I compared it and could not see the difference.

Yet my program failed and his program got full marks. I think I deserve these marks. Fitch-fork is wrong for not giving me my marks.

In this case this student made a non-significant formatting error which he did not recognise when he compared his solution with that of his fellow student. Unfortunately this error resulted in the automatic assessment system (Fitch-fork) awarding no marks. We acknowledge that the automatic assessment may have been unfair, but condemn the practice of substituting someone else’s code to remedy the problem.

4. STUDENT KNOWLEDGE

To determine how well the students are educated about plagiarism and its consequences, a question about this was included in a class test at the beginning of 2014. The class test date was published well in advance and the students were free to attend any one of three scheduled sessions for the test. Because these venues are the ones that are booked for normal classes, the students sit shoulder-to-shoulder in these venues when writing their test. We created different versions of the test. The questions were shuffled and changed. Different versions of the test were copied onto different coloured paper and handed out in a chequered manner to reduce opportunities for plagiarism while writing. The question about plagiarism was asked in only one version of the test and consequently answered by only 51 out of 511 (9.98%) students who wrote the test, which is 8.5% of the number of registered students at the time of writing the test. The question that was posed to them was:

Explain why committing plagiarism is not a good idea and discuss its implications.

The student answers revealed different themes from those that emerged on the discussion board. More detailed definitions were given and none of the answers could be classified as scenario descriptions. This can be attributed to the fact that students could give straight answers without fear of repeating something that was said by someone else as is the case on the discussion board. The students also elaborated much more on the consequences of committing plagiarism when compared with the extent to which consequences were mentioned on the discussion board, probably because they were prompted to do so. As with the discussion board, the answers were classified in categories. Figure 4 shows the number of answers that were classified in each of the categories. Note that the total does not add to the number of answers, because many answers were classified in more than one category. There were answers that mentioned all the aspects.

The following is an example of such an answer:

Plagiarism is not a good idea because by doing so you are not learning anything from the module and just receiving marks you do not deserve. You will also end up not mastering the module by yourself and won’t have any understanding. The implication is serious; since it is stealing someone’s work. This is an illegal action and can result in no marks for the specific assignment and may turn in expulsion that will ruin your future.

5. CONCLUSION

We have demonstrated that our students are confused about the definition of code plagiarism. This is no surprise as literature that addresses this issue is scarce and the concepts that apply to plagiarism in general have to be adapted in many ways when applied to code. It is virtually impossible to identify plagiarism by merely looking at the end product. The process applied while creating the end product is important in determining if the code is plagiarised or not. If a student applies some standard refactoring tricks to a solution that was created by someone else without understanding the code, it can be seen as plagiarism. If, however, a student would use existing code as a template to write his/her own code to create a solution to a task that is similar to the example, it may be seen as an illustration that he/she understands the example and can apply it to the new situation. In such a case it may not be seen as plagiarism. Only the person who created the code will know if the work that was done can be classified as a learning experience or as an action to score undeserved marks.

If a student submits unaltered code that was created by someone else for a task that was given as an individual task, it is undeniably plagiarism and can be detected by the most basic similarity checks if the original code is available. More sophisticated code plagiarism detection software is able to identify similarity of code where simple refactorings like changing variable names, rephrasing comments, changing the order of function definitions, and the like, are applied. When the original source is not available or when students are aware of the tell-tale signs of plagiarised code, plagiarism detection software may be unable to identify plagiarism, but that does not change the fact that the code is plagiarised.

Our observation and experiences when applying punishment in order to address plagiarism appears to be successful. Plagiarism detection software usually only quantifies the similarity between different documents according to certain metrics. Thereafter it still requires an expert to judge if it is plagiarism or not. If the similarity is 100% when a metric like the time stamp of creation of a document is taken into account, it does not take much judgement to be able to decide. The more specific a detection metric is in the identification of copying, the more likely it becomes that cases of plagiarism are not detected and the easier it is to fool the detection algorithm. Our detection strategy scratched only the surface. The students soon discovered how to avoid detection.

The analysis of the test answers provided by our students shows that they have theoretical knowledge of the concept of plagiarism and are acutely aware of the legal consequences of being caught. They are, however, not able to apply this general concept to define code plagiarism. Some students mentioned ethical reasons why plagiarism is unacceptable and may be aware that this act may rob them of an opportunity to learn. The prominence of referring to punishment as a consequence, how-
ever, indicates that many students are still biased towards immediate outcomes and results rather than towards concerted effort to meet learning goals.

We concur with Wagner [23] who states that it is important to eradicate plagiarism one way or the other because it undermines learning. Like Shaw [21] we are convinced that the prevention of cheating is better than to expend effort in establishing that cheating has occurred and taking action against students who did indeed plagiarise. Prevention entails reducing opportunities to cheat, minimising temptations to cheat as well as fulfilling our educational responsibility to create a learning culture and instil ethical values. Kourie [13] emphasises that curricula should aim to transmit an agreed-upon value-system, using skills training as a context to achieve this. Lecturers should constantly reinforce values relating to professionalism, responsibility, ethics, etc. This can be done by maximising opportunities to assimilate these values. The core message that should be brought to our students is aptly described in the words of Scott Alexander:

*All good is hard. All evil is easy. Dying, losing, cheating, and mediocrity is easy. Stay away from easy* [3].

6. REFERENCES


the 17th ACM annual conference on Innovation and technology in computer science education, ITiCSE ’12, pages 122–127, New York, NY, USA, 2012. ACM.


