Student Perspectives on Plagiarism in Computing

Mike Joy[1], Jane Sinclair[1], Russell Boyatt[1], Jane Yin-Kim Yau[2] and Georgina Cosma[3]

[1]Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
[2]School of Computer Science, Physics and Mathematics, Linnaeus University, Sweden
[3]Department of Business Computing, PA College, Larnaca, Cyprus

**Abstract**

Prevention and detection of plagiarism have formed the basis for many research projects, but student perceptions on plagiarism are arguably not well understood, and this is particularly true in the computing disciplines. This paper considers two aspects of the student experience, (i) the types of plagiaristic activity that students engage in, and (ii) the specific understanding of what plagiarism means for students who write computer programs. In a recent study we classified types of plagiarism in computing, collecting data from two sources: published material (books, published papers, web sites), and on-line formative quizzes and questionnaires used by universities to test student knowledge of what constitutes plagiarism. Facet analysis was used to categorise the data into four initial categories (sources, actions, material, extrinsic). Further analysis suggested a refinement to six categories and 23 sub-categories which directly relate to the computing disciplines. In a further study we analysed data collected from 18 UK institutions via an online questionnaire, in which over 700 computing students were invited to scrutinise fifteen scenarios which may (or may not) contain plagiarism, and to identify which they thought did contain plagiarism. The results of this study suggest that there are certain types of plagiaristic activity which are poorly understood, and demographic factors which correlate with student understanding. This paper summarises and compares the results of these two studies and reflects on the implications for educating computing students about how they should avoid plagiarism.

## Introduction

Plagiarism in student assignments continues to be a major concern in universities (Roberts, 2007; Dey and Sobhan, 2006) and tools such as TurnitIn are now commonly used to assist in the detection process. Much of the research into plagiarism in recent years has concentrated on the detection aspects and particularly on the technical challenges of tool-based support (for example, Potthast *et al*., 2011). Increasing effectiveness of automated detection is seen as contributing to the higher observed rates of plagiaristic activity, although some studies suggest that the relationship between different factors is rather more complicated than this (Culwin, 2009). At the same time there has been continued interest in investigating students' perceptions and understanding of plagiarism, this being closely linked to students' reasons and motivations for submitting plagiarised work (Marshall and Garry, 2005; Power, 2009). A number of current research initiatives, such as IPPHEAE (http://ippheae.eu/) and The Citation Project (http://site.citationproject.net/) continue to investigate aspects of students' citation activity and their perspectives on plagiarism.

As with the majority of work so far, current initiatives focus in the main on text-based plagiarism. Much less has been done relating specifically to the computing disciplines and to the understanding of source code plagiarism. Although the basic concept of plagiarism is the same whether applied to text or code, there are some fundamental differences which distinguish consideration of the two. One area concerns the techniques, algorithms and tools that are effective for detection of plagiarism in each case and this has led to the development of a separate range of plagiarism detection tools for source code such as MOSS (Bowyer and Hall, 1999), JPlag (Prechelt *et al.*, 2002) and Sherlock (Joy and Luck, 1999). A further area in which differences may be observed is that of student perception. Although there has been little work on this, there are indications that students may be less likely to view the copying of code as an offence or that they may have greater confusion over what is classed as plagiarism in the context of code. For example, Mann and Frew (2006) discovered that students regarded a proportion of 60% to 90% similarity as "normal" – that is, insufficient for regarding as plagiarism. The situation is clearly more complicated for code because of legitimate similarity introduced by factors such as language structure, institution style and the use of code stubs in assignments. Mann and Frew's work also indicated that students were very unclear about acceptable levels of collaboration on program assignments. In a study by Chuda *et al.* (2012), students appeared to have a reasonable understanding of plagiarism but were uncertain as to whether code plagiarism in assignments is acceptable or not. Within a sample of 1149 students asked whether program plagiarism is "wrong", 1% replied that it was not and a further 69% did not respond. Within this group, 33% admitted to having plagiarised.

Chuda *et al*.'s study raises some interesting issues, such as the reasons why so many students did not understand source code plagiarism to be an academic offence and whether this was consistent with their views on text plagiarism. The questions asked were of a fairly general nature and did not test in detail whether students really did understand the concept of plagiarism by probing their understanding in different cases. Within text-based plagiarism it has been found that while there is a common consensus about what plagiarism essentially means, there is a "grey area" of activity which may or may not be considered as plagiaristic (Stolley and Brizee, 2010). We wish to explore this possibility for source code plagiarism. If such areas of confusion can be identified, student educational resources on source code plagiarism can be strengthened by including further material on the areas which most often cause misunderstanding.

This paper considers two aspects of the student experience: (i) the types of plagiaristic activity which computing students engage with, and (ii) the specific understanding of what plagiarism means for students who write computer programs. Drawing on the results of our two recent studies, we consider how a categorisation of plagiaristic activities can help us to understand how to deal with borderline plagiarism and how these match with student perception of those activities.

## Types of plagiaristic activity

In order to identify and classify different types of plagiaristic activity within the context of source code we first conducted a literature-based study and categorisation exercise (Joy *et al.*, 2009). A comprehensive classification of issues relating to source code plagiarism provides an empirical foundation for subsequent development of resources which can accurately assess a student's understanding of what plagiarism actually means and assist the student in avoiding plagiarism.

The study gathered data from two sources. The first consisted of on-line interactive resources, such as web sites published by institutions, which contain tests to measure students' awareness of plagiarism and provide feedback based on the students' responses. Twenty-three such resources were identified, which collectively contained 268 questions, and were located in 4 UK universities and 14 US universities and colleges.

The second data source was published books (including Carroll (2007), Decoo (2002) and Roberts (2008)) and published papers on plagiarism and academic misconduct. Of these, some specifically addressed source-code plagiarism, such as Culwin *et al*. (2001) and Cosma and Joy (2008).

The data collected (topics occurring as the subject of quiz questions or raised as issues in published sources) were analysed using Facet Analysis (Broughton, 2004; Lambe, 2007). Facet analysis allows items to be identified according to a number of different, independent aspects or categories. It represents a "bottom-up" approach to mapping a subject area, working from individual items to identify clusters of interest. We first applied the generic framework suggested within facet analysis which resulted in the identification of four relevant categories (*sources* of plagiarised material, the different *actions* of plagiarised activity, types of *material* involved, the *extrinsic* factors relating to context). A second level of analysis was applied to re-factor the possible categories, to identify combinations most relevant to source code plagiarism and to further subdivide these in a manner which could be directly aligned with the development of on-line educational and test material. This resulted in a final set of six categories with 23 sub-categories as shown:

1. Plagiarism and copying
   - 1.1. Ideas: referencing people's experiences, impressions, ideas and inspirations (which are not stored as a document which can be referenced)
   - 1.2. Facts: referencing commonly known facts, such as basic mathematical facts, common geographical and historical facts
   - 1.3. Speech: referencing someone saying something (e.g. referencing the words of a TV presenter in a documentary, a story told by a friend, or what was said while interviewing a friend)
   - 1.4. Copying: identifying what constitutes copied material (text, figures, images) from various sources of information, and which should be referenced.
   - 1.5. Paraphrasing: acceptably paraphrasing text or editing diagrams
   - 1.6. Self-plagiarism: referencing work that was previously submitted for academic credit (or publication)
   - 1.7. Avoidance strategies: good practice for avoiding plagiarism
   - 1.8. Translating text: translating text between languages
   - 1.9. Email: copying words from email, IM, or other personal contacts
2. Referencing
   - 2.1. Referencing: correctly referencing, placing quotation marks where appropriate, and citing in appropriate formats
3. Cheating and inappropriate collaboration
   - 3.1. Collaboration: identifying when it is acceptable for students (or groups of students) to collaborate and sharing work
   - 3.2. Purchasing: purchasing academic material such as essays or hiring experts to write essays or source code (contract cheating)
   - 3.3. Cheating: Other cheating issues (not necessarily called "plagiarism"), such as falsification and fabrication
4. Ethics and consequences
   - 4.1. Ethics: understanding the relevance of ethical behaviour, copyright and fair use related to plagiarism
   - 4.2. Consequences: consequences (punishments, etc.) when students are caught
5. Source code plagiarism
   - 5.1. Adapting source code: adapting (modifying) source code written by other programmers
   - 5.2. Open source: using and referencing Open Source code
   - 5.3. Copying source code: using and referencing source code written by other programmers
   - 5.4. Code generation: referencing source code which has been automatically generated
   - 5.5. Translating code: translating source code between programming languages including algorithms written in pseudo code or diagrams such as UML
6. Source code documentation
   - 6.1. Documentation: copying comments in source code or other documentation
   - 6.2. Designs: copying source code or interface design material and reverse engineering
   - 6.3. Testing: copying test data and/or test strategy

Of particular interest to this paper are categories (5) and (6).

**Students' perceptions on what constitutes plagiarism**

The second study (Joy *et al*., 2011) investigated students' understanding of source code plagiarism by presenting them with a series of scenarios. Data was collected from 770 computing students at UK and European institutions using an online questionnaire. Respondents were presented with 15 scenarios, each of which may or may not describe a plagiaristic activity relating to computer program source code. For each scenario, respondents were asked "Is this plagiarism?" and had to choose from a (Likert scale) of 5 responses: "Yes, definitely", "I think it is, but I am not completely sure", "I don't know", "I think it isn't, but I am not completely sure" and "No, definitely not". Each scenario was carefully examined by (at least) four academics each of whom was experienced in managing and detecting instances of plagiarism, and for each scenario all academics agreed unanimously that the correct response was either "Yes" or "No" (in other words, none of the scenarios was ambiguous). The students were also asked to provide demographic information, the university at which they were studying (optional), and whether they had been instructed

about plagiarism.

Of the 770 respondents, 77% chose to declare their university, and these consisted of 18 institutions in the UK (13 "pre-1992" and accounting for 68% of the respondents; 5 "post-1992" and accounting for 9% of respondents) and 3 in Europe (representing less than 1% of the respondents). 53.2% of respondents were undergraduates taking a BSc degree in a computing subject, 20.6% a taught MSc in a computing subject, 16% were studying a joint BSc degree in computing with another subject, 4.7% were research students in a computer discipline. 702 of the 770 respondents claimed to be European, and 37 were Asian.

Each question was marked 1, 0.5, 0, -0.5 or -1 (following the Likert scale described above), allowing for a "total mark" for each respondent in the range [-15, 15]. Here, 15 representing correct answers with full confidence in all scenarios. The frequencies of scores attained are illustrated in figure 1.

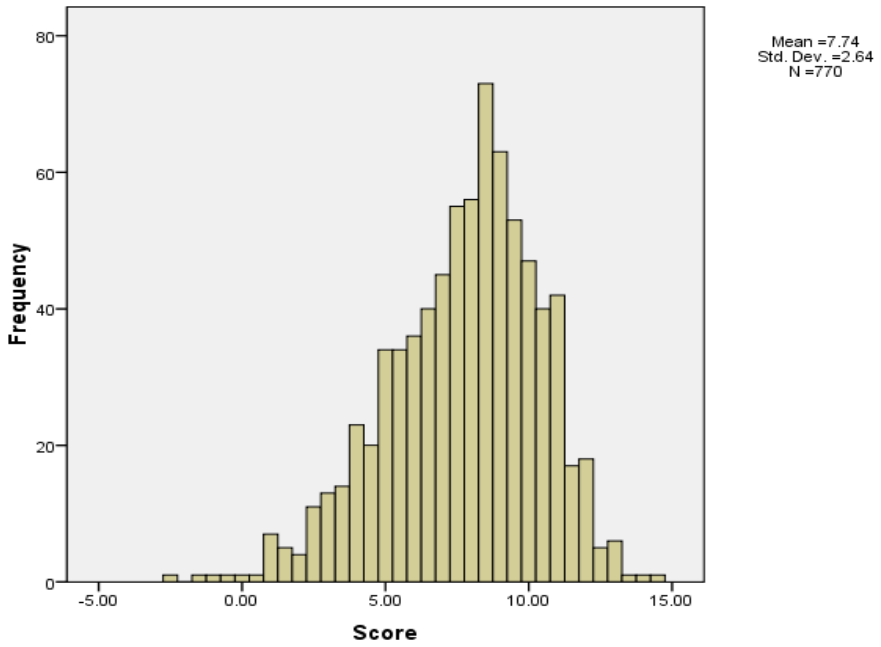


**Figure 1: Frequencies of total scores (from Joy *et al*., 2011)**

Hypotheses were tested that the scores obtained may correlate with student background (degree programme, type of university, demography), but the data generally did not support any such hypotheses.

The scenarios were divided into six topics. In order for the quiz to be at an appropriate length (so that respondents would engage meaningfully), and in order to focus on a particular set of hypotheses, not all categories and sub-categories derived in the first study were mapped to questions in the quiz.

Topic 1: Self-plagiarism and source-code re-use;
Topic 2: Copying code from books and other sources;
Topic 3: Copying code from another student;
Topic 4: Inappropriate collaboration;
Topic 5: Converting code to another programming language;
Topic 6: Falsification (as opposed to plagiarism).

Topic 1 contained two scenarios, and was motivated by lack of awareness by staff of re-use of previously submitted assignments constituting self-plagiarism, either source code (Cosma and Joy, 2008) or text (Marshall and Garry, 2005). In the first (1a), code from a previous assignment was re-used with an acknowledgement in a comment field, and in the second (1b), such code was incorporated without acknowledgement. Whilst almost all (90%) understood that the first was not plagiarism, only 7% thought that plagiarism might have taken place in the second. A t-test indicates strong negative correlation between the responses to the two scenarios ($t=-0.36$, $p<0.001$, $n=755$). The responses to these scenarios show that most students do not think re-using their own work constitutes plagiarism of any kind.

Topic 2 comprised 5 scenarios each involving code which has been copied from a book. In (2a) the source of the code is not acknowledged at all, but in (2b) the student claims to have forgotten where the code was copied from. In (2c) a group project is described with all references complete, and in (2d) the student claims consults (referenced) textbooks "for ideas" and "to gain inspiration" rather than sources for the code. In (2e), a group project contains notes in the comments to the effect that the code has been re-used from elsewhere. Over 90% of respondents correctly identified

(2a) as plagiarism and (2c) and (2e) as not plagiarism. However only 31% identified (2b) as plagiaristic, suggesting that the "correctness" of citations is not understood as being fundamental. Furthermore, only 77% considered that (2d) was acceptable, suggesting confusion as to the distinction between copying material and using ideas.

Topic 3 was well understood by almost all respondents. The three scenarios involved copying code from an unattended terminal (3a), paying a student in a previous year to author part of the code (3b) and copying the contents of a printout left in a waste basket. Both (3a) and (3c) were correctly identified as plagiarism by 97% and 92% respectively, but the correct response for (3b) of 86% suggests that there may be confusion as to the precise meaning of plagiarism (vis-à-vis cheating).

Topic 4 consisted of two scenarios: in (4a), two students work together on an individual assignment and submit very similar pieces of work, whereas in (4b), a group assignment, two students assigned to two *different* groups exchange code and submit it as part of their separate group submissions. Unexpectedly, although 62% thought (4a) was definitely plagiarism, 28% were unsure, and for (4b) 62% considered the collusion to be unacceptable but 31% were unsure. These figures suggest that a substantial proportion of students view working together on code to be acceptable even when explicitly informed otherwise. This finding ties in with the observation of Barrett and Malcolm (2006) that simply telling students is insufficient: students should be an active part of their own plagiarism education.

The two scenarios in topic 5 related to a C++ program taken from a textbook and converted to Java to be submitted as a student assignment without the textbook being mentioned (5a) and a Visual Basic program written by the student at school converted to a Java program when at university (5b), but the provenance of the code noted in a comment. Fewer than half the respondents (49%) recognised (5a) as plagiarism, but almost all (97%) correctly identified (5b) as not being plagiarism. Thus it appears that language translation of code is misunderstood.

Topic 6 consisted of one scenario, submitting a program which does not work correctly but displays the "correct" output. This is not plagiarism (although it is definitely fraudulent!), and 89% of students agreed.

**Discussion**

This paper summarises the results of two recent studies: one has produced a classification of issues relating to plagiarism (including source-code plagiarism) and the other has analysed student perceptions of source-code plagiarism to identify topics which are poorly understood. We can focus on the classification sub-categories to which the latter study relates, as summarised in Table 1.

| Category | Topic | Problem |
|---|---|---|
| 1.6: Self-plagiarism | 1 | Re-use of code submitted for previous assignments is not generally understood as plagiarism |
| 1.4: Copying<br>5.3: Copying source code | 2, 3 | Students are not sensitive to the importance of *correct* references for copied code |
| 3.1: Collaboration | 4 | Sharing code when an "individual" activity is mandated is often perceived as acceptable |
| 5.5: Translating code | 5 | Translating code is not fully recognised as plagiaristic activity |
| 3.3: Cheating | 6 | The distinction between plagiarism and other forms of cheating seems to be reasonably well understood in the context of computer programs |

**Table 1: Results of the student perception survey mapped against the categories of plagiarism issues**

The scenarios used in the second activity did not cover the whole range of possible plagiaristic activities but were constructed to represent a number of likely situations and to cover some areas which had been observed to cause confusion in practice. By using the categorisation from the first study we can map exactly which themes have been covered and which have not, constructing different resources to support areas as required. It also allows for a bank of different scenarios to be assembled in which scenarios are tagged with category information. Different variants of quizzes can then be generated as required to address specific aspects or to avoid students becoming familiar with the same scenario.

The second study demonstrates that, despite instruction on the "dos and don'ts", there are genuine areas of confusion which may cause students to misunderstand what is required. There may be others in addition to the five we have noted and further work is required with additional scenarios to elicit information on this. However, these five provide a good indication of areas which should be reinforced to students. Confusion as to the acceptable limits of collaboration is a particularly difficult problem in relation to source code. Students are often encouraged to help each other learn by discussing their work. They are also aware that in the "real world" system development is a collaborative activity – indeed, students with work experience may be used to proceeding in such a way. Further, some institutional policy is

couched in terms of discussing ideas being acceptable and collaborating on producing an assignment being unacceptable. Involvement in open-source programming projects or programming assignments designed as group-work may further develop this collaborative approach to software development. Another issue is the reproduction of implementation strategies and ideas rather than direct source-code plagiarism. For example, particularly complex structuring mechanisms or sequences of software library calls may indicate a level of idea plagiarism (as related to category 1.1). Students within a course or institution may also have very similar strategies for implementing common algorithms due to nature of instruction or teaching materials used. Yet these things are difficult to separate in practice and it is unsurprising that grey areas exist.

Some areas of confusion may be seen as less serious than others: for example, a self-plagiarism may be viewed as a minor misdemeanour in comparison with failing to acknowledge the source of a translated program. However, such activity may still to accusations of academic misconduct (for example, most institutions would not allow resubmission in whole or in part of work which has already been submitted for credit).

## Conclusion

We have used the results of two previous studies (Joy *et al.*, 2009; 2011) to focus on some of the issues which are viewed by educators (as evidenced through training materials) and academics (as reported in the literature) to be important facets of the plagiarism phenomenon. The second study has given us an insight into how computing students view plagiarism in the context of writing computer programs. Combining the two, we have identified five problems which, if addressed when educating students about plagiarism, might focus attention on common and significant misunderstandings.

Our work provides further insight into the issue of plagiarism in the specific area of program source code addressing in particular the extent to which students really understand the issues involved. There has often been debate over whether students understand what is required, whether plagiarism is intentional or unintentional, and whether educational resources concerning plagiarism are effective.

Previous studies revealed that a general consensus exists among educators that a zero tolerance plagiarism policy is appropriate; and thus, when reuse is permitted, students should adequately acknowledge the parts of the source-code written by other authors (or that the students have submitted as part of another assessment) otherwise these actions can be construed as plagiarism (or self-plagiarism) (Cosma and Joy, 2008).

The fact that source-code reuse is encouraged in object-oriented programming may lead some students to take advantage of this situation, and use source-code written by other authors without providing adequate acknowledgements. Educators teaching programming should inform students on source-code reuse and acknowledgement, so as to reduce the occurrences of plagiarism (Cosma and Joy, 2008).

Our work indicates that misunderstanding is likely to be a contributory factor. It also provides a practical way to structure educational resources and map students' understanding.

## References


Barrett, R. and Malcolm, J., 2006. Embedding plagiarism education in the assessment process. *International Journal. for Educational Integrity* **2**(1) pp. 38-45.

Bowyer, K. and Hall, L., 1999. Experience using MOSS to detect cheating on programming assignments. *29th Annual Frontiers in education conference FIE99*, San Juan, Puerto Rico, pp 18-22.

Broughton, V., 2004. *Essential Classification*. London: Facet Publishing, pp. 257-283.

Carroll, J., 2007. *A Handbook for Deterring Plagiarism in Higher Education (2nd edition).* Oxford Centre for Staff and Learning Development.

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

Cosma, G. and Joy, M.S., 2008. Towards a Definition on Source Code Plagiarism. *IEEE Transactions on Education* **51**(2) pp. 195-200.

Culwin, F., 2009. I think my students are less naughty, but maybe the tools are more effective? *Proceedings of the 2nd International Plagiarism Conferen*ce, Gateshead, UK, March 2009, pp.10-13.

Culwin, F., MacLeod, A. and Lancaster, T., 2001. *Source Code Plagiarism in U.K H.E Computing Schools, Issues, Attitudes and Tools*. Technical Report SBU-CISM-01-02, South Bank University, London.

Decoo, W., 2002. *Crisis on Campus: Confronting Academic Misconduct.* Cambridge, MA: MIT Press.

Dey, S. and Sobhan, A., 2006. Impact of unethical practices of plagiarism on learning, teaching and research in higher education: Some combating strategies. *Proceedings of the 7th International Conference on Information Technology Based Higher Education and Training*, pp. 388–393.

Joy, M.S., Cosma, G., Yau, J.Y-K. and Sinclair, J.E., 2011. Source Code Plagiarism - a Student Perspective. *IEEE Transactions on Education* **54**(1) pp. 125-132.

Joy, M.S., Cosma, G., Sinclair, J.E. and Yau, J.Y-K., 2009. A Taxonomy of Plagiarism in Computer Science. *Proceedings of the International Conference on Education and New Learning Technologies (EDULEARN09)*, Barcelona, Spain, 6-8 July 2009, pp. 3372-3379.

Joy, M.S. and Luck, M., 1999. Plagiarism in programming assignments. *IEEE Transactions on Education*, **51**(2), pp.129-133.

Lambe, P., 2007. *Organizing knowledge: Taxonomies, Knowledge and Organizational Effectiveness*. Oxford: Chandos Publishing.

Mann, S. and Frew, Z., 2006. Similarity and originality in code: plagiarism and normal variation in student assignments. *ACE '06 Proceedings of the 8th Australasian Conference on Computing Education*, volume 52, pp. 143-150.

Marshall, S. and Garry, M., 2005. How well do students really understand plagiarism? in *Proc. ASCILITE Conference*, Brisbane, 2005, pp. 457-467.

Potthast, M., Eiselt, A., Barron-Cedeno, A., Stein, B. and Rosso, P., 2011. Overview of the 3rd International Competition on Plagiarism Detection. *Notebook Papers of CLEF 2011*, Amsterdam, Netherlands.

Power, L., 2009. University students' perceptions of plagiarism. *Journal of Higher Education* **80**(6) pp. 643-662.

Prechelt, L., Malpohl, G. and Philippsen, M., 2000. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science* **8**(11), pp1016-1038.

Roberts, T.S., 2008. *Student Plagiarism in an Online World: Problems and Solutions*, Hershey, PA: IGI Global.

Stolley, K. and Brizee, A., 2010. Is it Plagiarism yet? *Purdue Online Writing Lab*, Purdue University. Available at: http://owl.english.purdue.edu/owl/resource/589/2/ [accessed: 26 March 2012].