International Journal for Educational Integrity

Student perspectives on source-code plagiarism

M.S. Joy
Department of Computer Science, University of Warwick, Coventry, UK
m.s.joy@warwick.ac.uk

J.E. Sinclair
Department of Computer Science, University of Warwick, Coventry, UK
j.e.sinclair@warwick.ac.uk

R. Boyatt
Department of Computer Science, University of Warwick, Coventry, UK
russell.boyatt@warwick.ac.uk

J.Y-K. Yau
Department of Computer Science, Malmö University, Malmö, Sweden
jane.yau@mah.se

G. Cosma
Department of Business Computing, PA College, Larnaca, Cyprus
g.cosma@pacollege.ac.cy

An earlier version of this paper was presented at the 5th International Integrity and Plagiarism Conference, Newcastle-upon-Tyne, UK, July 2012 (Joy, Sinclair, Boyatt, Yau & Cosma 2012).

Keywords: source-code plagiarism, computing, student perspectives, plagiarism taxonomy

Abstract

Prevention and detection of plagiarism has formed the basis of much research, but student perceptions on plagiarism are arguably not well understood. This is particularly the case 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 programmes. In a recent study, data were collected from published material (books, published papers, websites), and online formative quizzes and questionnaires used by universities to test student knowledge of what constitutes plagiarism. Facet analysis was used to classify 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 a large-scale online questionnaire was carried out to obtain the perceptions of students on source-code plagiarism. Data were collected from 770 students studying at 21 higher education institutions in the UK and overseas. This study's results suggest that certain types of plagiaristic activity are poorly understood. 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.

The International Journal for Educational Integrity is available online at:
http://www.ojs.unisa.edu.au/journals/index.php/IJEI/

Introduction

Plagiarism in student assignments continues to be a major concern in universities (Dey & Sobhan, 2006; Roberts, 2008), mainly because of students' inadequate understanding of actions that constitute plagiarism and failure to comprehend and practice appropriate citation techniques (Gullifer & Tyson, 2010; Marshall & Garry, 2005). It is important that academics educate students on the topic to eliminate instances of unintentional plagiarism, and for them to take action when they suspect misconduct in order to improve student behaviour and reduce instances of intentional plagiarism (Nadelson, 2007).

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 & Brizee, 2010). This paper explores this possibility for source-code plagiarism by identifying areas most confusing to students and improving educational resources to tackle these issues. It considers two aspects of the student experience: (i) the types of plagiaristic activity which computing students engage in, and (ii) the specific understanding of what plagiarism means for students who write computer programmes. It draws upon the results of two previous studies (Joy, Cosma, Sinclair, & Yau, 2009; Joy, Cosma, Yau, & Sinclair, 2011).

Although academics may suspect plagiarism, collating enough evidence to convince the relevant academic panel in charge of dealing with plagiarism cases can be a difficult and time demanding activity (Joy & Luck, 1999; Larkham & Manns, 2002). To assist this process, the detection stage often involves using plagiarism detection tools, such as *Turnitin*, to detect similarities between student assignments and articles found on the internet. 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, work by Potthast, Eiselt, Barron-Cedeno, Stein and Rosso (2011).

A number of studies (such as Carter, 2000 and Sheard, Markham, & Dick, 2003) have investigated IT students' attitudes to and reasons for cheating. The specific focus of the current research is plagiarism (rather than more general cheating and deception) and, while there are undoubtedly many students who knowingly choose to plagiarise, studies consistently indicate that lack of awareness and understanding is a major factor causing a significant number of students to unwittingly break the rules, as noted by Park (2003).

Students' perceptions and understanding of plagiarism are closely linked to their reasons and motivations for submitting plagiarised work (Marshall & Garry, 2005; Power, 2009). A number of current research initiatives, such as the Impact of policies for plagiarism across Europe project (IPPHEAE) (Glendinning, 2012) and The Citation Project (Howard, Rodrigue & Serviss, 2010) continue to investigate aspects of students' citation activity and their perspectives on plagiarism.

Research into student perceptions reveals that confusion on plagiarism-related matters still exists despite the development of good practice guides (such as Carroll & Appleton, 2001) and the efforts of institutions and instructors to provide information and instil good practice. Students are often unclear about what constitutes intentional cheating and many display poor citation practice (Gullifer & Tyson, 2010).

A recent survey by Egaña (2012) provided further evidence for students' lack of understanding concerning the importance of citing and referencing in their assignments. Importantly, they found that students often do not cite the sources of information they use because their lecturers do not specifically request them to do so.

Many students fear being penalised if they provide references as this would indicate that they have used ideas and information created by someone else.

Wilkinson (2009) found the most common reasons for plagiarism to be lack of understanding of referencing rules, bad time management, and easy access to material over the internet. The same study identified contradicting views between academic staff and students as to whether students receive adequate guidance concerning acceptable referencing. Seventy-eight percent of staff believed guidance to be adequate, as opposed to 57% of students. The opinion that students do not understand the rules of referencing was shared by both academics and students.

Issues for plagiarism in general also apply to assignments involving the development of computer programmes. There are reasons why there may be even greater confusion over what is acceptable when source-code is involved. Programmers normally use existing software libraries to build new software, rather than building everything from scratch – hence, source-code re-use is acceptable practice by professional programmers who aim not to 're-invent the wheel'. For this reason, source-code re-use may be encouraged in some programming assignments (after all, the purpose is to prepare students for employment). It is therefore important for university policies to address acceptable software re-use, plagiarism and ethics in software development.

Some research indicates that students are indeed less likely to view the copying of source-code as an offence or that they may have greater confusion over what is classed as plagiarism in the context of source-code as opposed to text. Mann and Frew (2006) discovered that students regarded a proportion of 60% to 90% similarity in code 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 programme assignments. Similarly, Chuda, Navrat, Kovacova and Humay (2012) found that students appeared to have a reasonable understanding of plagiarism but were uncertain as to whether source-code plagiarism in programming assignments is acceptable. These studies raise some interesting issues, such as why so many students did not understand source-code plagiarism to be an academic offence, whether this was consistent with their views on text plagiarism, and whether there were specific areas of confusion.

Another aspect differentiating text-based and source-code plagiarism concerns the techniques, algorithms and tools that are effective for detection of plagiarism in each case. This has led to the development of a separate range of plagiarism detection tools for source-code such as MOSS (Bowyer & Hall, 1999), JPlag (Prechelt, Malpohl, & Philippsen, 2002), Sherlock (Joy & Luck, 1999), and a recently developed cloud application, 'CodeAliker', which automates the submission of assignments and the plagiarism detection process of essay text and computer code (Upreti and Kumar, 2012).

Gibson (2009) formulated a code of practice for software re-use which covers all the documents produced during the engineering of a software system – analysis, requirements, validation, design, verification, implementation and testing. Included examples demonstrate acceptable and unacceptable forms of software re-use. The proposed code of practice also addresses issues concerning reverse engineering, unacknowledged translation, unacknowledged code generation and software testing. However, despite such initiatives, it seems that a large amount of confusion still exists amongst students concerning source-code plagiarism and that plagiarism education programmes are not currently succeeding in clarifying acceptable practice.

The first study reported here presents a taxonomy of plagiaristic activities which can be utilised by academics to develop appropriate resources for educating students on source-code plagiarism issues. The proposed taxonomy can be used when testing students' understanding on source-code plagiarism and in devising questions tailored to students' needs. For example, the taxonomy could be used to develop questions that test understanding of source-code plagiarism and citation of code in a particular programming language, and to map questions to all or part of the areas identified under the source-code category. Instructors may also include more questions on areas which they believe are more problematic to their students.

The second study uses the proposed taxonomy to design a survey purposed to gather student perceptions on source-code plagiarism and to identify source-code plagiarism areas confusing to students. The survey findings may help academics understand how to deal with source-code plagiarism and borderline plagiarism.

Types of plagiaristic activity

To identify and classify different types of plagiaristic activity within the context of source-code, a literature-based study and categorisation exercise was conducted (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 them in avoiding plagiarism.

The study gathered data from two sources. The first consisted of online interactive resources, such as websites published by institutions, which contained 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. These were located in 4 UK universities and 14 US universities and colleges.

The second data source consisted of 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, MacLeod and Lancaster (2001) and Cosma and Joy (2008).

Data collected on 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 is an approach to knowledge organisation, allowing items to be identified according to a number of different, independent aspects or categories ('facets'). It describes an examination of the domain of interest, identifying themes which are independent and mutually exclusive, and covering all the topics. Determining such a classification involves a degree of subjective consideration but there are guidelines which outline areas to be considered.

These guidelines were followed by four independent subject experts, who each identified a list of mutually exclusive facets which were then discussed to reach agreement on appropriate top-level categories. Initially, application of the generic facet analysis framework 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).

The main motivation for this categorisation exercise was to support an online educational resource. As such, the relative importance of the four facets (or the topics to which they relate) are not equal. For example, a computing student is unlikely to translate musical lyrics obtained from a radio station, but is much more likely to copy

source-code. Furthermore, it would be impractical to use all permutations of the four facets as classifications of questions in an educational tool.

A second level of analysis was therefore 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 online educational and test material. As originally described in Joy et al. (2009), 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 share 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.

Different categorisations of the topic could be produced; however, the above list gives one possible approach which provided a good starting point for investigating students' understanding. 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. Respondents were presented with 15 scenarios, each of which may or may not describe a plagiaristic activity relating to computer programme source-code. Existing published sources, public online quizzes and the authors' experience of student behaviour were used as a basis for identifying areas relating to source-code plagiarism which are regarded as important and often misunderstood topics. Scenarios were generated relating to each of these topics as described below. The topics chosen for the scenarios were codified according to the taxonomy described above. This activity had two motivations: it allows the exact topics covered by a particular quiz to be tracked, mapping which areas have been addressed and which have not. It allows misunderstandings confirmed by testing to be codified in a similar way. Further tests or banks of questions can be developed according to areas the author wishes to cover or which students are shown to find difficult. It would be impractical to try to cover every possible category in a single quiz, and indeed this may well be unnecessary since some areas may not generally cause confusion.

For each scenario, respondents were asked "Is this plagiarism?" and were required 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". In order to ensure that none of the scenarios was ambiguous, each scenario was carefully examined by (at least) four academics, each of whom was experienced in detecting and managing instances of plagiarism. For each scenario all academics agreed unanimously that the correct response was either "Yes, definitely" or "No, definitely not". Following standard practice, the survey was piloted with a group of students prior to distribution to ensure clarity both of the questions and of the order in which the questions were presented.

The students were also asked to provide demographic information, the name of the university at which they were studying (optional), and whether they had been instructed about plagiarism. Data were collected from 770 computing students. Almost all of the respondents were studying at higher educational institutions located in the UK or the European Union (702), with the next largest grouping, Asia, containing just 37. The origins of the remaining 31 respondents included Africa (7), North America (5), the Middle East (4), and Australasia (1). Of the 770 respondents, 77% chose to declare their university's name. These consisted of 18 institutions in the UK (13 "pre-1992" accounting for 68% of the respondents, 5 "post-1992" accounting for 9% of respondents) and 3 in Europe (representing less than 1% of the respondents). Of the respondents, 53.2% were undergraduates taking a BSc degree in a computing subject, 20.6% were enrolled in a taught MSc in a computing subject, 16% were

studying a joint BSc degree in computing with another subject and 4.7% were research students in a computer discipline. 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 represents correct answers with full confidence in all scenarios. Although Likert scales are not interval scales, this approach gave a clear description of student responses, the mark for each question indicating the 'distance' from the correct answer (which was in each case a clear "definitely plagiarism" or "definitely not plagiarism"). The frequencies of scores attained are illustrated in Figure 1.

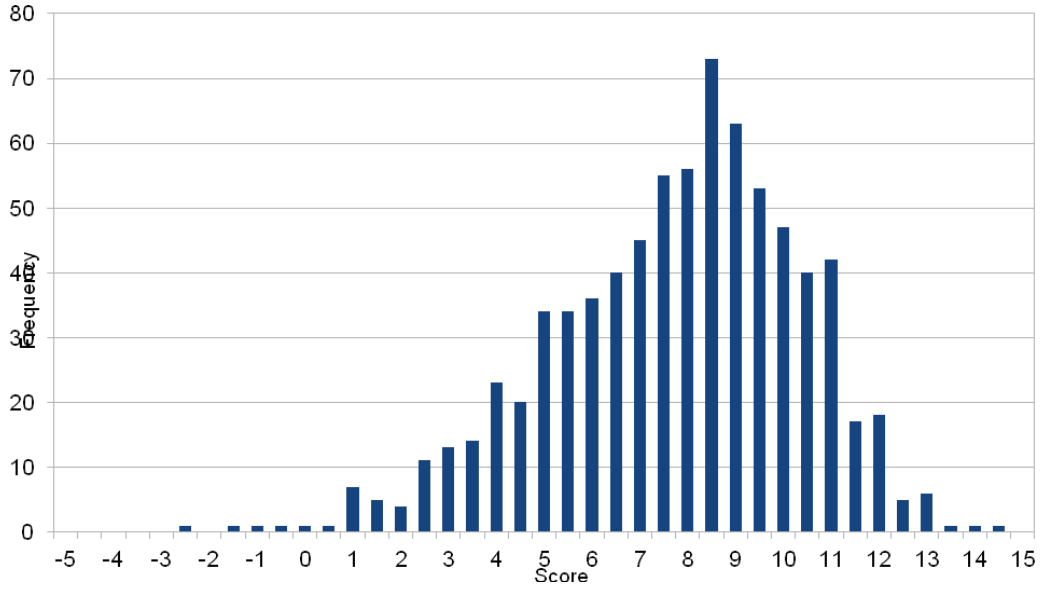

Figure 1. Frequencies of total scores

The hypothesis was that scores obtained might correlate with student background (degree programme, type of university or demography), and t-tests were used to test possible correlations, however no significant demographic correlation was found.

Previous studies (such as Sheard et al., 2003) suggest that there is significantly less plagiarism at the postgraduate level, but the reasons remain unclear. Such studies do not distract from the current results, which focus on uncovering areas of confusion. It is perfectly consistent to claim that MSc students tend to be confused about the same aspects as undergraduates (even if they do plagiarise less).

The scenarios were divided into six topics:

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).

For the quiz to be a reasonable length (so respondents 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. In particular, only

those directly related to source-code plagiarism were addressed. In the following discussion, the 15 scenarios are reproduced verbatim.

Topic 1 contained two scenarios, and was motivated by lack of awareness concerning the re-use of previously submitted assignments constituting self-plagiarism, either when source-code (Cosma & Joy, 2008) or text (Marshall & Garry, 2005) was concerned. Re-using parts of a programming assignment previously submitted for academic credit, and incorporating these into another assignment without providing adequate acknowledgement of this fact is considered to be self-plagiarism, and addressed as such by university policies. The two scenarios were:

> (1a) Whilst writing a large Java program required for his third year project, Tom decides to re-use some source-code which he had authored previously. He acknowledges this in his program in the form of a comment, and submits it as his own work.
>
> (1b) Last term, John was rewarded high marks for a C++ programming assignment he authored and submitted. This term, as part of his final year project, he wants to use exactly one of the functions he previously created for his C++ assignment. He finds the function, takes it and incorporates it into his completed final year project and submits it.

Whilst almost all (90%) understood that scenario 1(a) was not plagiarism, only 7% thought that plagiarism might have taken place in scenario 1(b). 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. This finding revealed that perceptions of self-plagiarism between students and academics vary. In an earlier survey by Cosma and Joy (2008) which analysed perceptions of academics on source -code plagiarism, 81% of academics viewed re-using one's own source-code in a different programming assignment without acknowledgement as unacceptable and a plagiaristic activity. The following table summarises the responses for Topic 1. Shaded cells indicate the correct answer for each scenario.

Table 1:
Responses for Topic 1

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No re-sponse	Total responses
1 (a)	9	9	3	55	**670**	9	746
1 (b)	**26**	25	8	81	612	3	752

Topic 2 comprised five scenarios involving code which had been copied from a book. The scenarios presented to students were as follows:

> (2a) Andy is required to submit a Java applet program for his assignment. He remembers reading about a similar applet program from a textbook he's been using. He goes to find this, and uses this code and then submits it as his own program without noting that he obtained the code somewhere else.
>
> (2b) Amy writes down some source-code from a text-book in the library which she wants to re-use as part of her C++ program, with the intention of making an acknowledgement of it in her program. She then returns to the computer lab but has forgotten to write down the name of the book. She then searches on the

library website and writes down the name of another Java book and makes a reference to this book in her program, and then submits it.

(2c) Charlie and his group have completed their software engineering group project and just need to make references to the program code and their design of classes where they have re-used others' material. They then note in the program in the form of comments the re-use of code, and in their documentation the re-use of classes.

(2d) Samantha is writing some source-code on a particular class for her C++ program. In order to help her gain some inspirations on the code she wants to write, she uses both a C++ and a Java textbook to read for ideas. She then authors and completes her program, and submits it as her own work without referencing the textbooks.

(2e) Robin and his project group have finished their final year group project dissertation and are ensuring that:

- all their names are included on the project cover,
- code authored by each individual member is commented,
- any other type of re-use in their program is commented, and
- all references are complete.

They then submit their dissertation.

Over 90% of respondents correctly identified (2a), where source of the code is not acknowledged, as plagiarism, and (2c), where a group project is described with all references complete and (2e), where a group project contains notes in the comments indicating the code has been re-used from elsewhere as not plagiarism. Surprisingly, only 31% identified (2b), where the student claims to have forgotten where the code was copied from, as plagiaristic, suggesting that the "correctness" of citations is not understood as being fundamental. The action described in (2b), which involved giving a false citation, is considered as plagiarism and falsification. Falsification is the action of giving a false reference; that is, a reference that exists but does not match the copied source-code. Falsification is a form of cheating and co-occurs with plagiarism (since no citation is provided to the copied source-code) (Cosma & Joy, 2008). Regarding scenario (2d), where the student consults the textbooks "for ideas" and "to gain inspiration", only 77% correctly considered it as acceptable, suggesting confusion as to the distinction between copying material and using ideas. (2d) is not considered as plagiarism, and there is general consensus among academics that taking ideas or inspiration from code produced in the same or another programming language and creating the source-code entirely "from scratch" is not considered as plagiarism (Cosma & Joy, 2008). The following table summarises the responses for Topic 2. Shaded cells indicate the correct answer for each scenario.

Table 2:
Responses for Topic 2

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No response	Total responses
2 (a)	**613**	95	11	23	12	1	754
2 (b)	**89**	145	114	235	165	7	748
2 (c)	13	22	26	111	**578**	5	750
2 (d)	73	71	32	209	**367**	3	752
2 (e)	4	5	5	33	**704**	4	751

© International Journal for Educational Integrity Vol. 9 No. 1 June, 2013 pp. 3–19 ISSN 1833-2595

Topic 3 was well understood by almost all respondents. The three scenarios presented to students were as follows:

> (3a) Nick and Paul are working on the same Prolog assignment in the computer lab, and are sitting next to each other. Paul briefly leaves his seat without locking his computer. Nick glanced over and saw on Paul's computer some completed code for the assignment, and copies it before Paul comes back. Nick then makes minor adaptations of this to his program, and submits it as his own work.
>
> (3b) Fred knows James who is in the year above him, and offers to pay him a small amount of money for creating a part of his program for him. James agrees and authors part of the program for him. Fred then completes the program, and submits it as his own work.
>
> (3c) Whilst collecting a printout from the printer, David sees a printed program which has been left uncollected for a long time. He takes the uncollected printout along with his own printout, and copies the work from the uncollected printout to his program, and submits this.

Both (3a), copying code from an unattended terminal and (3c), copying contents of a printout left in a waste basket, were correctly identified as plagiarism by 92% and 97% respectively. The correct response of plagiarism for (3b), paying a student in a previous year to author part of the code, was chosen by 86% suggesting there may be confusion over the precise meaning of plagiarism (as opposed to other forms of cheating). Submitting another author's code for credit, either with permission (such as someone else willingly producing the work as in scenario (3b)) or without permission of the original author (for example, stealing, cheating as occurs in scenarios (3a) and (3c)) constitutes plagiarism. Verbatim copying and altering source-code authored by someone else without providing acknowledgement are also considered as plagiarism (Cosma & Joy, 2008). The following table summarises the responses for Topic 3. Shaded cells indicate the correct answer for each scenario.

Table 3:
Responses for Topic 3

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No response	Total responses
3 (a)	**636**	58	16	28	13	4	751
3 (b)	**587**	57	27	43	38	3	752
3 (c)	**673**	55	12	9	5	1	751

Topic 4 consisted of two scenarios:

> (4a) Steve decides that it would be more efficient to work on his Java applet programming assignment together with his friend on the same module. The assignment requires that they work alone. They then submit very similar Java applets.
>
> (4b) Russell and Antony are both required to work on their software engineering group projects, however they have been assigned different groups to work in. They decide to help each other out anyway by exchanging some parts of the assignment which they have problems with and are unsure about. They both then include the exchanged work in their group projects and submit them.

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 plagiarism, 28% were unsure, and for (4b) 62% considered the collusion to be unacceptable but 31% were unsure. Collusion occurs when students collaborate on an assignment which requires students to work individually. The results indicate that a substantial proportion of students view working together on code to be acceptable even when explicitly informed otherwise. Barrett and Malcolm (2006) have noted that simply informing students of plagiarism policy is not sufficient: they must play an active role in their own plagiarism education. Although academics consider it acceptable and 'pedagogically valuable' for students to share ideas and discuss their assignments, submitting similar assignment solutions due to close collaboration or collusion is considered unacceptable. Sharing ideas and sharing solutions are two very different activities with the latter resulting in an academic offence (Cosma & Joy, 2008). The following table summarises the responses for Topic 4. Shaded cells indicate the correct answer for each scenario.

Table 4:
Responses for Topic 4

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No response	Total responses
4 (a)	**254**	213	61	126	95	6	749
4 (b)	**232**	231	71	126	92	3	752

The two scenarios in topic 5 related to source-code taken from one language and translated into another:

> (5a) Whilst writing some source-code on a particular class for her C++ program, Sarah remembers the particular class being in a Java textbook that she was reading. She goes to obtain this Java textbook and converts the Java class from the textbook into C++ and adapts it to her program, and then submits it as her own work without referencing that textbook.
>
> (5b) Sophie has previously written a Visual Basic program for her A-level project, and wants to incorporate a function from there to her first year Java program. She first converts the function to Java, and then incorporates it, and notes this fact in her program in the form of 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. The scenarios (5a) and (5b) concern the action of taking source-code from one programming language and directly converting it line-by-line to a different but similar programming language. (5a) involves copying and converting someone else's programming solution without providing acknowledgement, and this action is perceived as plagiarism in academia (Cosma & Joy, 2008). Concerning (5b), the student avoided committing self-plagiarism by providing an appropriate comment. The following table summarises the responses for Topic 5. Shaded cells indicate the correct answer for each scenario.

© International Journal for Educational Integrity Vol. 9 No. 1 June, 2013 pp. 3–19 ISSN 1833-2595

Table 5:
Responses for Topic 5

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No response	Total responses
5 (a)	**186**	179	45	201	140	4	751
5 (b)	5	8	10	49	**673**	10	745

Topic 6 consisted of one scenario, submitting a programme which does not work correctly but displays the "correct" output.

> (6a) Eric is working on his Java program and manages to make it compile and run, however it does not produce the wanted outputs that are required for this week's assignment. He doesn't understand why it's not producing the wanted outputs and he decides that in order to solve this problem, he would just modify the program output to make it produce the wanted output, whilst the rest of the program was not functioning as the assignment requires it to. He then submits this work.

Modifying the program output to make it seem as if the program works when it does not is a form of academic offence akin to falsification and may co-occur with plagiarism (Cosma & Joy, 2008). However, (6a) is not plagiarism since it did not involve someone else's work, and 89% of students agreed. The following table summarises the responses for Topic 6. The shaded cell indicates the correct answer for the scenario.

Table 6:
Responses for Topic 6

Scenario	Yes, definitely	I think it is	I don't know	I think it isn't	No, definitely not	No response	Total responses
6 (a)	22	19	40	71	**596**	7	748

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. The classification sub-categories to which the latter study relates are summarised in Table 7.

Table 7

Results of the student perception survey mapped against the categories of plagiarism issues

Category	Topic	Problem
1.6: Self-plagiarism	1	Re-use of code submitted for previous assignments is not generally understood as plagiarism.
1.1: Ideas 5.3: Copying source-code	2, 3	The distinction between copying material and using ideas is confusing to students. The correctness of citations for copied code is not understood as being fundamental by students.
3.1: Collaboration	4	Sharing code when only "individual" activity is mandated is often incorrectly 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 in the context of computer programmes is confusing to some students.

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. As noted in Table 7, although the first five areas were indeed confirmed as causing problems, the sixth (the difference between plagiarism and other forms of cheating) seemed to be well understood by most students. 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. The mapping shown is not limited to source-code only sub-categories (from categories 5 and 6). The problems identified can be linked to more general plagiarism issues such as self-plagiarism and collusion, demonstrating that some issues occur irrespective of the format. The mapping allows for a bank of different scenarios to be assembled, each scenario being 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 appear to be genuine areas of confusion which can cause students to misunderstand what is required. There may be others in addition to the five noted in Table 7 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. In the UK, a recent report from the Office of the Independent Adjudicator refers to a doubling of reported plagiarism cases since 2008 and states that there is much that still needs to be done in setting out policy and explaining expectations to students (OIA, 2011). The study provides evidence of existing misunderstandings and suggests areas to be clarified.

Of particular interest are the scenarios from Topic 4 concerning collaborative activities. Confusion as to the acceptable limits of collaboration is a particularly difficult problem in relation to source-code, since students are often encouraged to help each other learn by discussing their work. Discussions may include the sharing of programming techniques, information on library functions, and the general approaches to a problem. In some institutions, policy is couched in terms of

discussing ideas being acceptable and collaborating on producing an assignment being unacceptable, and students may find this hard to interpret in practice. Students are also aware that in the 'real world' system development is a collaborative activity – indeed, students with work experience might be used to proceeding in such a way. This may be another possible explanation for students' lack of clarity on acceptable limits of cooperation. Collaborative system development in the software industry is common practice and re-use of source-code within an organisation would be encouraged to save both time and money. This approach is at odds with the individual approach necessary for assessment purposes. 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 the nature of instruction or teaching materials used. Students learning common algorithms for routine tasks such as searching and sorting would be taught to implement these as clearly and efficiently as possible and be expected to use these throughout their software development career. While minor details such as variable names and formatting are inevitable, the basic structural components for a given programming language are likely to be very similar. It would be undesirable for students to deviate from well-known and understood implementation strategies for common algorithms simply to avoid accusations of plagiarism. For common algorithms a high degree of similarity is not only expected but may also be encouraged. Identifying exactly what constitutes plagiarism can therefore be difficult and it is unsurprising that grey areas exist for students and instructors alike.

Some areas of confusion may be seen as less serious than others: for example, 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 result in 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

The results of two previous studies (Joy et al., 2009; 2011) have been used to focus on some of the issues which are viewed by educators as being important aspects of plagiarism. The second study has provided an insight into the perceptions of computing students on plagiarism, in the context of writing computer programmes. Combining the two studies has revealed five specific problems which focus attention on common and significant misunderstandings, and should be addressed when educating students about plagiarism. They may also have implications for institutional policy and the need to clarify what the stated rules mean in practice. The discussion in this paper strongly reinforces the guidance that, for each assignment given to students, instructors must work to ensure that their students have a good understanding of the meaning of plagiarism as it relates to that specific item of coursework, and will need to clarify issues such as the difference between the (allowable) sharing of ideas and (plagiaristic) collusion. Below we summarise the five 'grey areas' identified in this study.

The first problem is student awareness of self-plagiarism. There is a common misunderstanding that it is acceptable to re-use code which has already been submitted for a piece of coursework in a subsequent assignment. When assignments permit source-code re-use, 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).

Secondly, institutional policy often differentiates between copying and adapting another person's code (unacceptable) and obtaining ideas and inspirations from others (acceptable). However, the dividing line between gathering ideas from external sources and inappropriately copying may be unclear to students (and indeed to staff as well).

Thirdly, copying source-code may be acceptable but only if appropriate acknowledgements or references are provided. Providing false or fake acknowledgements is also an academic offence which co-occurs with plagiarism – an incorrect reference could be a simple mistake but it might also be a deliberate attempt to hide an original source from which the code has been copied. How appropriate references should be provided in the case of source-code is often unclear. There is genuine uncertainty caused by the encouragement of source-code re-use in object-oriented programming, and furthermore, since software solutions include producing designs and testing-related artefacts to accompany source-code, such additional materials can also be plagiarised.

Fourthly, the desirability of student collaboration outside of individual assessments (reinforced by the generally collaborative nature of 'real' software development) sends mixed messages to students and the 'individualness' of individual assignments can easily be misunderstood.

Finally, translating code from one language to another without acknowledgement is plagiarism, since the solution and structure are copied, but this is also poorly understood by students.

Repeated studies have revealed that although academics consider it essential for students to cite their sources correctly, many do not encourage good referencing practices in their lectures. In particular, programming instructors should inform students clearly of their expectations, especially concerning source-code re-use and acknowledgement. The results of this survey have revealed five areas of possible student misunderstanding relating to source-code plagiarism, and this information will assist instructors when informing students about how to avoid plagiarism.

References

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

Bowyer, K., & Hall, L. (1999). Experience using MOSS to detect cheating on programming assignments. *29th Annual frontiers in education conference FIE99,* San Juan, Puerto Rico, 18–22.

Broughton, V. (2004). *Essential classification*. London: Facet Publishing.

Carroll J., & Appleton, J. (2001). Plagiarism: A good practice guide, JISC report.

Carroll, J. (2007). *A handbook for deterring plagiarism in higher education* (2nd ed.). Oxford Centre for Staff and Learning Development.

Carter, J. E. (2000). What the students said about plagiarism. *Proceedings of ITiCSE 2000*, Helsinki, ACM Press.

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

Cosma, G., & Joy, M. S. (2008). Towards a definition on source code plagiarism. *IEEE Transactions on Education*, *51*(2), 195–200.

Culwin, F., MacLeod, A., & Lancaster, T. (2001). *Source code plagiarism in U.K H.E computing schools, issues, attitudes and tools*. Technical Report SBU-CISM-01 -02. London: South Bank University.

Decoo, W. (2002). *Crisis on campus: Confronting academic misconduct.* Cambridge, MA: MIT Press.

Dey, S., & 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*, 388–393.

Egaña, T. (2012). Use of bibliography and academic plagiarism among university students. *Universities and Knowledge Society Journal (RUSC)*, *9*(2), 200–212.

Gibson, P. J. (2009). Software re-use and plagiarism: A code of practice. *In Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE '09), ACM,* New York, NY, USA, 55–59.

Glendinning, I. (2012). European responses to student plagiarism in higher education, *Proceedings of the 5th International Integrity and Plagiarism Conference,* Newcastle-upon-Tyne, UK, July 2012.

Gullifer, J., & Tyson, G. A. (2010). Exploring university students' perceptions of plagiarism: A focus group study. *Studies in Higher Education*, *35*(4), 463–481.

Howard, R. M., Rodrigue, T.K. & Serviss, T.C. (2010). Writing from Sources, Writing from Sentences. Writing and Pedagogy, 2(2), 177-192.

Joy, M. S., Sinclair, J. E., Boyatt, R., Yau, J. Y-K., & Cosma, G. (2012). Student perspectives on plagiarism in computing. *Proceedings of the 5th International Integrity & Plagiarism Conference*, Gateshead, UK, July 2012.

Joy, M. S., Cosma, G., Yau, J. Y-K., & Sinclair, J. E. (2011). Source code plagiarism: A student perspective. *IEEE Transactions on Education*, *54*(1), 125–132.

Joy, M. S., Cosma, G., Sinclair, J. E., & 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, 3372–3379.

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

Larkham, P., & Manns, S. (2002). Plagiarism and its treatment in higher education. *Journal of Further and Higher Education*, *26*(4), 339–349.

Lambe, P. (2007). *Organizing knowledge: Taxonomies, knowledge and organizational effectiveness*. Oxford: Chandos Publishing.

Mann, S., & 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*, 52, 143–150.

Marshall, S., & Garry, M. (2005). How well do students really understand plagiarism? *Proceedings of the ASCILITE Conference*, Brisbane, 457–467.

Nadelson, S. (2007). Academic misconduct by university students: Faculty perceptions and responses. *Plagiary*, *2*(2), 1–10.

OIA. (2011). Office of the independent adjudicator. *Annual Report.* Retrieved March 26, 2012, from www.oiahe.org.uk.

Park, C. (2003). In other (people's) words: Plagiarism by university students – literature and lessons. *Assessment & Evaluation in Higher Education*, *28*(5), 471–488.

Potthast, M., Eiselt, A., Barron-Cedeno, A., Stein, B., & 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), 643–662.

Prechelt, L., Malpohl, G., & Philippsen, M. (2002). Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, *8*(11), 1016–1038.

Roberts, T. S. (2008). *Student plagiarism in an online world: Problems and solutions*. Hershey, PA: IGI Global.

Sheard, J., Markham, S., & Dick, M. (2003). Investigating differences in cheating behaviours of IT undergraduate and graduate students: The maturity and motivation factors. *Higher Education Research & Development*, *22*(1), 91–108.

Stolley, K., & Brizee, A. (2010). Is it plagiarism yet? *Purdue Online Writing Lab*, Purdue University. Retrieved March 26, 2012, from http://owl.english.purdue.edu/owl/resource/589/2/.

Upreti, N., & Kumar, R. (2012). 'CodeAliker': Plagiarism detection on the cloud. *Advanced Computing: An International Journal (ACIJ)*, *3*(4), 21–26.

Wilkinson, J. (2009). Staff and student perceptions of plagiarism and cheating. *International Journal of Teaching and Learning in Higher Education 2009*, *20* (2), 98–105.

About the authors

Mike Joy is an Associate Professor in the Department of Computer Science at the University of Warwick in the UK, having received his PhD in computer science from the University of East Anglia. His research interests focus on educational technology and computer science education, with a specific focus on automatic assessment and plagiarism detection.

Jane Sinclair received her PhD in Computer Science at the Open University, UK. She is currently an Associate Professor in the Department of Computer Science at the University of Warwick. Her interests include educational technology, computer security and formal methods.

Russell Boyatt is a Research Associate in the University of Warwick's Computer Science Department where he is also completing his PhD. His research interests include educational technology, software development and formal methods and he is currently working on the EU Lifelong Learning Project 'MALog', developing learning resources for mathematical logic.

Jane Yau is a Postdoctoral Researcher at Malmö University in Sweden. Having graduated with a PhD from the University of Warwick she has recently been working at Linnæus University. Her interests include computer science education and mobile technologies.

Georgina Cosma is currently a Lecturer at PA College in Cyprus. She received a PhD in Computer Science from the University of Warwick. Her main research interests include educational technology, information retrieval, sentiment analysis and neural networks.