Towards a Definition of Source-Code Plagiarism

Georgina Cosma and Mike Joy

***Abstract*—A survey using a scenario-based questionnaire format has provided insight into the perceptions of U.K academics who teach programming on computing courses. This survey across various higher education (HE) institutions investigates what academics feel constitutes source-code plagiarism in an undergraduate context. Academics' responses on issues surrounding source-code reuse and acknowledgement are discussed. A general consensus exists among academics that a "zero tolerance" plagiarism policy is appropriate; however, some issues concerning source-code reuse and acknowledgement raised controversial responses. This paper discusses the most important findings from the survey and proposes a definition of what can constitute source-code plagiarism from the perspective of U.K. academics who teach programming on computing courses.**

***Index Terms*—Reuse, self-plagiarism, source-code plagiarism definition, survey.**

I. Introduction

PLAGIARISM in programming assignments is an issue for most academics teaching programming. Source-code can be obtained in various ways including the Internet, source-code banks, and text books. Online resources exist where students can hire expert coders to implement their programming assignments [1]. These opportunities make plagiarism easier for students. Concerns about the ease with which students can obtain material from online sources and use the material in their student work have been expressed in a number of journals, including [2] and [3].

Dick *et al.* and Sheard *et al.* identify various student cheating techniques and reasons why students cheat [4], [5]. In addition, they define cheating in the form of questions that, if answered positively, could indicate cheating. Culwin *et al.* conducted a study of source-code plagiarism in which they obtained data from 55 U.K. higher education (HE) computing schools [6]. They found that 50% of the 293 academics who participated in their survey believed that in recent years plagiarism has increased. Asked to estimate the proportion of students undertaking source-code plagiarism in initial programming courses, 22 out of 49 staff responded with estimates ranging from 20% to more than 50%. Decoo's recent book on academic misconduct discusses various issues surrounding academic plagiarism, and briefly discusses software plagiarism at the level of user-interface, content and source-code [7]. However, in order to identify cases of plagiarism one must have a clear idea of what actions constitute plagiarism. Sutherland-Smith completed a survey to gather the perspectives of teachers in the faculty of Business and Law at South-Coast University in Australia [8]. The findings reveal varied perceptions on plagiarism among academics teaching the same subject, and the author suggests that a "collaborative, cross-disciplinary rethinking of plagiarism is needed." In addition, a review of the current literature on source-code plagiarism reveals a lack of research on the issue of what is considered source-code plagiarism from the perspective of academics who teach programming on computing courses.

This paper discusses the findings of a survey carried out to gather the perspectives of U.K. academics of what is understood to constitute source-code plagiarism in an undergraduate context. The responses revealed that a wide agreement exists among academics on the issue of what can constitute source-code plagiarism. Because of the object-oriented nature of some programming languages, some academics have identified important issues concerned with source-code reuse and acknowledgement (including self-plagiarism). They also noted differences between the approach to plagiarism adopted for graded and nongraded work. In the final section of the paper, the survey findings are used to suggest a definition of what can constitute source-code plagiarism from the perspective of academics who teach programming on computing courses.

II. Methodology

An online questionnaire was distributed to academics across U.K. HE institutions. The mailing list of academics was supplied by the Higher Education Academy Subject Centre for Information and Computing Sciences (HEA-ICS). The United Kingdom has approximately 110 HE level computing departments, for which the HEA-ICS attempts to provide support. The mailing list contained contact information for 120 academics, most of whom were assumed to have expertise in teaching programming.

The survey instructions specified that only academics who are currently teaching (or have previously taught) at least one programming subject should respond. The survey was completed anonymously, but a section in which the academics could optionally provide personal information was included. A total of 59 responses were received, of which 43 provided the name of their academic institution. These 43 academics were employed at 37 departments in 34 different institutions; 31 were English universities and 3 were Scottish universities. Responses received from two or more academics from the same institution were consistent with each other.

The questionnaire was comprised mostly of closed questions requiring multiple-choice responses. The majority of questions were in the form of small scenarios describing various ways students have obtained, used, and acknowledged material. The respondents were required to select, from a choice of responses, the type of academic offense (if any) that in their opinion applied to each scenario. Gathering the comments of academics on

Manuscript received December 30, 2006; revised May 24, 2007 and June 1, 2007.

The authors are with the Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: g.cosma@warwick.ac.uk; m.s.joy@warwick.ac.uk).

Digital Object Identifier 10.1109/TE.2007.906776

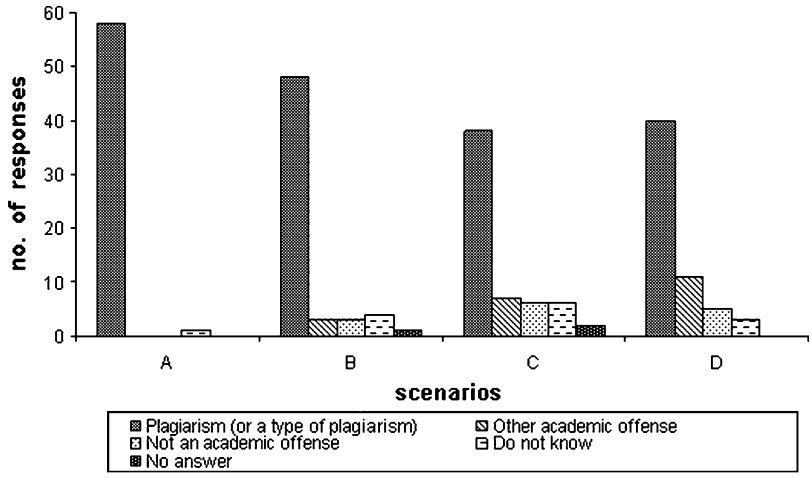

Fig. 1. Scenarios and responses. A: A student reproduces/copies someone else's source-code without making any alterations and submits it without providing any acknowledgements. B: A student reproduces/copies someone else's source-code, adapts the code to his/her own work and submits it without providing any acknowledgements. C: A student converts all or part of someone else's source-code to a different programming language and submits it without providing any acknowledgements. D: A student uses code-generating software (software that one can use to automatically generate source-code by going through wizards) and removes the acknowledgement comments that were automatically placed into the code by the software and submits it without providing any acknowledgements.

the various issues regarding plagiarism was important because of the variety both of university regulations in this area, and of the academics' opinions on such a sensitive issue. Therefore, a text-box was included below each question for academics to provide any comments they had about issues related to the question asked. A detailed analysis of the responses to all survey questions is reported elsewhere [9]. The purpose of this survey was not to address in depth subjective issues, such as plagiarism intent and plagiarism penalties, that could depend on student circumstances and university policies.

III. Survey Results

This section discusses academics' responses on issues surrounding source-code reuse and acknowledgement, assignment contribution, actions when plagiarism is suspected, and student collaboration in programming assignments. In the sections that follow, the term *module* denotes a single subject of study; for example, the *Programming for Computer Scientists module* is part of the undergraduate degree course (program) *Computer Science*.

A. Plagiarized Material

Plagiarism in programming assignments can go beyond the copying of source-code; it can include *comments*, *program input data*, and *interface designs*.

Comments within source-code can be plagiarized and may contribute towards identifying source-code plagiarism cases.

Program input data and the *user-interface* can be subject to plagiarism if they form part of the requirement in the assignment specification. The majority of respondents (40 out of 59) agreed that program input data can be subject to plagiarism but this alone cannot contribute to the identification of plagiarism. Three academics commented that copying input data is an issue if students are assessed on their testing strategies. When students are assessed on their testing strategies, assessment for plagiarism would occur by observing the testing strategy, including the datasets (e.g., input data) used for testing the program, and the testing material, including the test plan, system design documentation, technical documentation, and user manuals.

Interface designs submitted by students that appear suspicious need to be investigated for plagiarism if the assignment requires students to develop their own interface designs.

B. Adapting, Converting, Generating, and Reusing Source-Code

Academics were provided with scenarios concerned with the copying, adapting, and converting of source-code from one programming language to another, and using code-generating software for automatically creating source-code. A code-generator is an application that takes as input metadata (e.g., a database schema) and creates source-code that is compliant with design patterns. An example of shareware code-generator software is JSPMaker [10]—when given a database this software quickly and easily creates complete source-code and a full set of JavaServer pages [11] that have database connectivity.

The given scenarios and responses are shown in Fig. 1. Academics' comments on these scenarios raise important issues that are unique to source-code plagiarism.

Responses to scenarios A and B indicate a wide agreement that *reproducing/copying someone else's source-code with or without making any alterations and submitting it without providing any acknowledgements* constitutes source-code plagiarism. However, concerns were expressed on source-code reuse and acknowledgement. One academic commented

> "... in O-O environments where reuse is encouraged, obviously elements of reuse are not automatically plagiarism. I think I'd be clear on the boundaries and limits in any given circumstance, and would hope to be able to communicate that clarity to my students, but obviously there will potentially be problems. Use of the API would be legitimate without acknowledgement—or with only the implicit acknowledgement."

Regarding scenario B, many of the academics commented that adapting source-code may constitute plagiarism depending on the degree of adaptation, i.e., how much code is a copy of someone else's work and the extent to which that code has been adapted without acknowledgement. For example, a program may not be considered to be plagiarized if it was started by using existing source-code and then adapted to such an extent that it is beyond all recognition, so that there is nothing left of the original code to acknowledge. However, more of the respondents have raised the issue of source-code reuse and acknowledgement. Specifically, one academic remarked

> "... code copied from a website that assists in a specific task is potentially good practice. However, code that is a 100% copy is a different issue. I would also be concerned about the context of this copying. If the only deliverable were to be code and documentation the offense is clear. In this sense I suppose it is an issue of how much of the overall assignment is actually a copy of other work (without acknowledgement)."

On the issue of converting *all or part of someone else's source-code to a different programming language, and submitting it without providing any acknowledgements* (scenario C), several academics remarked that if the code is converted automatically without any or much effort from the student, then this procedure can constitute plagiarism. However, if a student takes the ideas or inspiration from code written in another programming language and creates the source-code entirely "from scratch," then this procedure is not likely to constitute plagiarism. Furthermore, in their comments academics have pointed out that taking source-code written in one programming language and converting it to a *similar* programming language, such as from C++ to Java, can constitute plagiarism. One academic, referring to scenarios A–D described in Fig. 1, emphasized the following:

> "In each case there must be some presumed benefit to the student in doing so (why did they do it otherwise?) and disruption to the assessment system. Even where the advantage might be minimal—e.g., from Prolog to C—a good student would almost certainly acknowledge the issue and use it to discuss the differences."

Academics were asked whether plagiarism takes place if "*a student uses code-generating software, removes the acknowledgement comments that were automatically placed into the code by the software, and submits it without providing any acknowledgements.*" The majority of the respondents considered unacknowledged use of code-generating software as plagiarism unless permission for use of such software is described in an assignment specification.

The findings suggest that students should be required to acknowledge any material they use that is not their own original work even when source-code reuse is permitted. All material should be acknowledged regardless of licensing permissions (e.g., open source, free-use, fair-use).

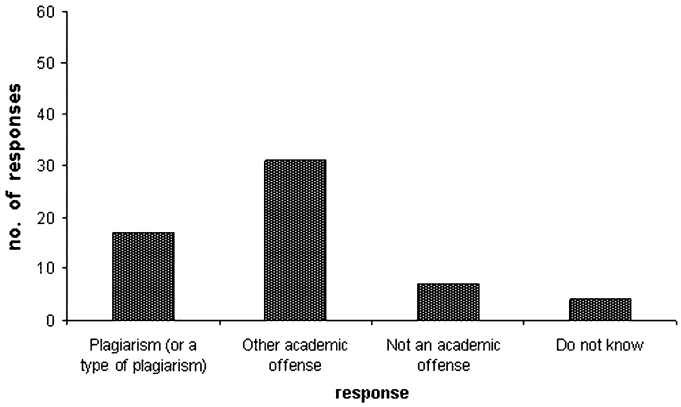

Fig. 2. Responses to self-plagiarism scenario. Scenario: Assume that students were not allowed to resubmit material they had originally created and submitted previously for an other assignment. For a graded assignment, a student has copied parts of source-code that s/he had produced for another assignment without acknowledging it.

C. Self-Plagiarism in Source-Code

In nonprogramming assignments, self-plagiarism occurs when a student reuses parts of an assignment previously submitted for academic credit and submits it as part of another assignment without providing adequate acknowledgement of this fact. In programming modules where source-code reuse is taught, self-plagiarism may not be considered as an academic offense.

Academics were given the scenario "*assume that students were not allowed to resubmit material they had originally created and submitted previously for another assignment. For a graded assignment, a student has copied parts of source-code that s/he had produced for another assignment without acknowledging it,*" and were asked to select a response. The results are shown in Fig. 2. Some controversial responses were received.

The majority of academics (48 out of 59) have characterized this scenario as an academic offense (17 as plagiarism and 31 as other academic offense), and in their comments they described this scenario as "self-plagiarism," "breach of assignment regulations if resubmission is not allowed," and "fraud if resubmission is not acknowledged." Some academics argued that in object-oriented environments, where reuse is encouraged, it is inappropriate to prevent students from reusing source-code produced as part of another programming assignment. The comments and responses provided by the academics who do not consider the given scenario to constitute plagiarism or another academic offense point to the controversial issue on source-code reuse mentioned previously. One academic who provided a "do not know" response remarked that students should reuse source-code where possible, while another, who was clear that the given scenario "was not an academic offense," emphasized:

> "I find it hard to assume that students were not allowed to resubmit material."

A third academic, who also stated that the given scenario "was not an academic offense," asked rhetorically

> "Would this ever happen in a programming-oriented module when we behove students not to reinvent the wheel?"

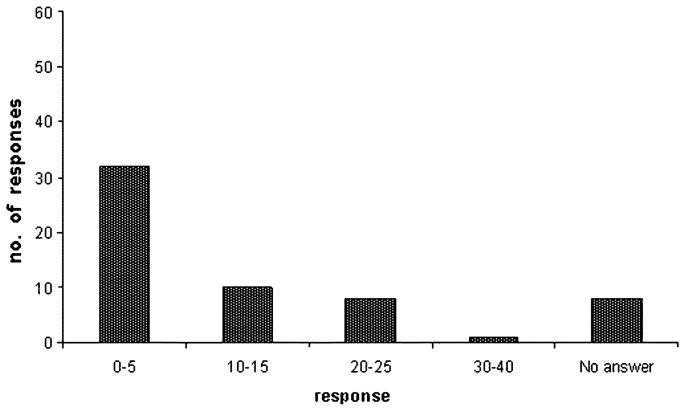

Fig. 3. Responses to minimum assignment weight scenario. Scenario: What would have to be the minimum weight of an assignment towards the overall module mark for you to proceed with investigation into plagiarism?

These statements represent the majority of surveyed academics, reinforcing our conclusion that resubmitting source-code without providing appropriate acknowledgements may lead to an academic offense, if this is not allowed for the particular assignment.

D. Assignment Contribution

When prompted to consider the question "*What would have to be the minimum weight of an assignment towards the overall module mark for you to proceed with investigation into plagiarism?*" responses were largely uniform. As Fig. 3 illustrates, the majority (32 out of 51) or respondents cited a value in the range of 0–5, suggesting a "zero tolerance" policy. Furthermore, there was agreement that investigation into possible plagiarism cases should always be pursued and appropriate action taken (either in the form of penalties or as warnings) regardless of the assignment's contribution towards the overall module mark.

The rationale for this attitude was clearly articulated by one respondent who warned:

> "If they cheat on a 5% assignment they will cheat on a larger one. A short, sharp shock can prevent the full offense."

In some instances, the severity of the offense is considered by the institution to vary depending on the contribution of the assignment:

> "Any contribution would bring the same investigation. However, for the full range of penalties to be applied a minimum contribution to the degree award of 7%, or a second or subsequent upheld accusation, is required. This is University policy."

Plagiarism can occur regardless of whether an assignment is graded or nongraded, i.e., regardless of its contribution to the overall course mark. Nongraded assignments refer to assignments that are not submitted for academic credit (such as some laboratory and tutorial exercises). Some university regulations on plagiarism seem to apply to graded assignments only. Many academics have commented that when plagiarism is detected in nongraded assignments, the students should be approached and informed or warned about its implications on graded assignments. Such measures are not always explicitly described in university regulations.

One academic commented that

> "If the assignment is not contributing towards the mark for the module then the correct protocol should be brought to the attention of the student."

E. Considerations on Student Collaboration

The survey raised issues on appropriate and inappropriate collaboration between students. There was general agreement among respondents that it is "pedagogically valuable" for students to engage actively in sharing ideas while discussing assignments as long as they do not copy each other's work. Three academics who encourage students to share ideas commented on the benefits and pitfalls of such practice:

> "I have no problem with sharing ideas. Given the restricted types of programs undergraduate students write, it is inevitable that certain design decisions lead to similar code."

This position was endorsed and elaborated by another academic who articulated a straightforward view on the problem:

> "I am personally not too worried about students discussing their work, it is the outright copying that I find most offensive and educationally pointless."

A third academic observed that it is

> "... important to remember that we often expect people to engage actively in discussing content and assignments. Also, as preparation for future working in teams, we often need to be much clearer in what we require of students—and what we don't."

Respondents noted that occurrences of plagiarism vary (both in type and in frequency) depending on the tasks being undertaken at the time. Respondents reported that occurrences of plagiarism when students are testing and debugging their software appeared to depend on the type of testing being carried out by the students. They remarked that occurrences of plagiarism during white box testing (tests requiring access to the code of the program under test) tend to be more frequent than during black box testing (tests conducted at the software interface level) due to the nature of these tests.

In addition, respondents noted that the distribution of marks awarded for the components of an assignment influences in which of those component tasks plagiarism may occur. For example, if the credit awarded for the design of a program is relatively high compared to the credit for the coding, then students are more likely to plagiarize when performing the design task.

Sharing ideas and sharing work were considered as two very different issues. Although academics expressed no objections to students sharing ideas, they opposed the practice of students collaborating and submitting similar work when assignments required them to work individually.

IV. Source-Code Plagiarism: Towards a Definition

Based on the responses summarized previously, the following is suggested as a new definition of what constitutes source-code plagiarism in an academic context.

Source-code plagiarism in programming assignments can occur when a student *reuses* (Section IV-A) source-code authored by someone else and, intentionally or unintentionally, fails to *acknowledge it adequately* (Section IV-C), thus submitting it as his/her own work. This involves *obtaining* (Section IV-B) the source-code, either with or without the permission of the original author, and *reusing* (Section IV-A) source-code produced as part of another assessment (in which academic credit was gained) without adequate acknowledgement (Section IV-C). The latter practice, *self-plagiarism*, may constitute another academic offense.

A. Reusing

"Reusing" includes the following:

1) reproducing/copying source-code without making any alterations;
2) reproducing/copying source-code and adapting it minimally or moderately; minimal or moderate adaptation occurs when the source-code submitted by the student still contains fragments of source-code authored by someone else;
3) converting all or part of someone else's source-code to a different programming language may constitute plagiarism, depending on the similarity between the languages and the effort required by the student to do the conversion; conversion may not constitute plagiarism if the student borrows ideas and inspiration from source-code written in another programming language and the source-code is entirely authored by the student;
4) generating source-code automatically by using code-generating software; this could be construed as plagiarism if the use of such software is not explicitly permitted in the assignment specification.

Where source-code reuse is not allowed, reusing (Section IV-A) source-code authored by someone else (or produced by that student as part of another assessment) and providing acknowledgements may constitute a breach of assignment regulations, rather than plagiarism (or self-plagiarism).

B. Obtaining

Obtaining the source-code either with or without the permission of the original author includes the following:

1) paying another individual to create a part of or all of their source-code;
2) stealing another student's source-code;
3) collaborating with one or more students to create a programming assignment which required students to work individually, resulting in the students submitting similar source-codes; such inappropriate collaboration may constitute plagiarism or *collusion* (the name of this academic offense varies according to the local academic regulations);
4) exchanging parts of source-code between students in different groups carrying out the same assignment with or without the consent of their fellow group members.

Incidents of source-code plagiarism can co-occur with other academic offenses (such as theft, cheating, and collusion) depending on academic regulations. The list previously mentioned is indicative of key areas where this form of plagiarism occurs, but it is certainly not exhaustive, since there are numerous ways that students can obtain source-code written by others.

C. Inadequately Acknowledging

Inadequately acknowledging source-code authorship includes the following:

1) failing to cite the source and authorship of the source-code, within the program source-code (in the form of an in-text citation within a comment) and in the appropriate documentation;
2) providing fake references (i.e., references that were made-up by the student and that do not exist); this is a form of academic offense, often referred to as *fabrication*, which may co-occur with plagiarism;
3) providing false references (i.e., references exist but do not match the source-code that was copied); another form of academic offense, often referred to as *falsification*, which may co-occur with plagiarism;
4) modifying the program output to make it seem as if the program works when it is not working; this too is a form of academic offense akin to falsification, which may co-occur with plagiarism.

V. Conclusion

Much survey-based research exists addressing the prevalence of source-code plagiarism in academia. However, surveys on the issue of what constitutes source-code plagiarism in U.K. universities are rare in academic scholarship. In addition, there appears to be no commonly agreed description of what constitutes source-code plagiarism from the perspective of academics who teach programming on computing courses.

Differences among university policies, assignment requirements, and personal academic preferences, can create varied perceptions among academics and students on what constitutes source-code plagiarism. The fact that source-code reuse is encouraged in object-oriented programming may lead students to take advantage of this situation, and use or adapt source-code written by other authors without providing adequate acknowledgements.

Since reuse is encouraged in object-oriented programming, some academics have expressed different opinions on issues surrounding source-code reuse and acknowledgement. The majority of respondents agreed that, 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).

General agreement exists that a "zero tolerance" plagiarism policy should be implemented. Responses show that university

policies influence the actions academics can take when they detect plagiarism. However, not all universities apply these policies to assignments that are not submitted for academic credit.

Academics teaching programming should inform students clearly of their preferences especially on source-code reuse and acknowledgement. Avoiding confusion among academics and students is likely to reduce the occurrences of plagiarism. Carroll and Appleton have devised a good practice guide suggesting techniques for dealing with plagiarism [12].

This paper considers the difference in opinions among academics on source-code specific issues and proposes a definition of source-code plagiarism, which can be adjusted by academics to meet their requirements.

ACKNOWLEDGMENT

The authors would like to thank the academics who responded to the survey, N. Griffiths for his contribution towards the creation of the survey questions, and N. Nakariakova and the staff at HEA-ICS for logistical support in managing the survey. Preliminary findings from the survey were reported in [13].

REFERENCES

[1] T. Jenkins and S. Helmore, "Coursework for cash: The threat from on-line plagiarism," in *Proc. 7th Annu. Conf. Higher Education Academy Network for Information and Computer Sciences*, Dublin, Ireland, Aug. 29–31, 2006, pp. 121–126.

[2] J. Kasprzak and M. Nixon, "Cheating in cyberspace: Maintaining quality in online education," *Assoc. Adv. Comput. Educ.*, vol. 12, no. 1, pp. 85–99, 2004.

[3] P. M. Scanlon and D. R. Neumann, "Internet plagiarism among college students," *J. Coll. Student Devel.*, vol. 43, no. 3, pp. 374–385, 2002.

[4] M. Dick, J. Sheard, C. Bareiss, J. Carter, D. Joyce, T. Harding, and C. Laxer, "Addressing student cheating: Definitions and solutions," *SIGCSE Bull.*, vol. 35, no. 2, pp. 172–184, 2003.

[5] J. Sheard, A. Carbone, and M. Dick, "Determination of factors which impact on IT students' propensity to cheat," in *Proc. 5th Australasian Computing Education Conf.*, Adelaide, Australia, 2003, pp. 119–126.

[6] F. Culwin, A. MacLeod, and T. Lancaster, "Source code plagiarism in UK HE computing schools, issues, attitudes and tools," South Bank Univ., London, U.K., Sep. 2001, Tech. Rep.

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

[8] W. Sutherland-Smith, "Pandora's box: Academic perceptions of student plagiarism in writing," *J. Eng. Acad. Purp.*, vol. 4, no. 1, pp. 83–95, 2005.

[9] G. Cosma and M. Joy, "Source-code plagiarism: A UK academic perspective," Dept. Computer Science, University of Warwick, Conventry, U.K., Res. Rep. No. 422, 2006.

[10] Computer Software, JSPMaker v1.0.1, e.World Technology Ltd., Hong Kong, China.

[11] H. Bergsten, *JavaServer Pages*, 3rd ed. Sebastopol, CA: O'Reilly, 2003.

[12] J. Carroll and J. Appleton, *Plagiarism: A Good Practice Guide*. Oxford, U.K.: Oxford Brookes Univ., 2001.

[13] G. Cosma and M. Joy, "Source-code plagiarism: A UK academic perspective," in *Proc. 7th Annu. Conf. Higher Education Academy Network for Information and Computer Sciences*, Dublin, Ireland, Aug. 29–31, 2006, pp. 116–120.

Georgina Cosma is working towards the Ph.D. degree in the Department of Computer Science, University of Warwick, Coventry, U.K.

Her fields of interest include source-code similarity detection and the latent semantic analysis information retrieval technique.

Mike Joy received the M.A. degree in mathematics from Cambridge University, Cambridge, U.K., the M.A. degree in postcompulsory education from the University of Warwick, Coventry, U.K., and the Ph.D. degree in computer science from the University of East Anglia, Norwich, Norfolk, U.K.

He is currently an Associate Professor at the University of Warwick. His research interests focus on educational technology and computer science education.