Nyinkeu, N. D., Thaddeus, K., & Henry, N. (2016). Open-source and attitude towards code-plagiarism among technology students. *International Journal of Technology in Teaching and Learning, 12(2*), 99-111.

Open-Source and Attitude towards Code-Plagiarism Among Technology Students

Ngatchu Damen Nyinkeu
Catholic University Institute of Buea, Cameroon

Kabuin Thaddeus
ICT University, Los Angeles

Ngatchu Henry
University of Buea, Cameroon

In an era when its is increasingly encouraged to share source-codes, the training of students, particularly information technology students, faces several challenges in communicating aspects of academic honesty – precisely authenticity of authorship. The genuineness of this issue has implications primarily in the integral formation of students with further implications in businesses which could face lawsuits due to a wrong attitude of one employee. This qualitative research explores an issue which is commonly overlooked and misconstrued even in academic circles. The findings unveil that today's technology students are more inclined to an open and flexible notion of code-plagiarism. The paper highlights a critical implication to the Technological Pedagogical Content Knowledge (TPACK) framework and concludes with recommendations for university ethics boards as well as possible future research directions.

Keywords: Plagiarism, Code-plagiarism, Educational System, Open-source, Sub Saharan Africa

INTRODUCTION

Open-Source is a term commonly used to describe computer software for which the original instructions of the software are made freely available and may be modified and redistributed. These instructions, written in plain text are known as the source codes of a software. Software developers and programmers fondly refer to the process of writing these instructions as coding and it involves converting ideas and concepts into text which computers can interpret and execute. Most software development companies would guard

---

Ngatchu Damen Nyinkeu is an Assistant Lecturer at Catholic University Institute of Buea in Cameroon. Kabuin Thaddeus is a Lecturer at ICT University at Baton Rouge, Los Angeles. Ngatchu Henry is an Assistant Lecturer at University of Buea in Cameroon. Ngatchu Damen Nyinkeu can be reached at ngatchu@cuib-cameroon.net.

their source-codes jealously, as intellectual property, since giving it out is tantamount to giving out the core of the business. Proponents of Open-source software argue that source code should be made readily available; a controversy which is rooted in the software licensing policy.

The General Public License, also known as the GNU General Public License (GNU GPL) and the Berkeley Source Distribution (BSD) license style are two initial open source licensing schemes that emerged. The underpinning difference resides in what can be done with products that are derived from open source software (oss). Both licenses grant broad rights to create derivative works but the BSD License grants these rights free and clear, while the GPL attaches important conditions (Gomulkiewicz, 2004). Over the years, a range of licensing variants have emerged notable the Apache license from apache software foundation, the Apple Public Source License (APSL) from apple incorporation, the Mozilla Public License (MPL) and the MIT License from Massachusetts Institute of Technology. The concept of copyleft is now perpetuated as a strategy for using copyright law to pursue the policy goal that prevents privatization of open-source codes but fosters and encourages the right to copy, share, modify and improve creative works of authorship (Goss, 2007).

The proliferation of open source products (Shimel, 2012) such as Android – the operating system for most mobile devices; Joomla and Wordpress – content management systems for building websites and Openstack – a cloud computing software used by NASA, HP, AT&T and Deustch Telekom is a clear indication of the success of open source projects. In the last decade, the technology giant – Google has invested millions of dollars in a summer activity called Google Summer of Codes (GSOC). GSOC is a global program that offers post-secondary student-developers aged 18 and above, a stipend to write codes for various open source software projects (Google Developers, 2016). According to statistics on the GSOC website, in 2005, a first cohort of students 400 students from 49 different countries participated in projects hosted in 40 different open source organizations. Remarkably, in 2014, 1307 students from 72 countries were accepted into the program to participate in 190 open source projects. The increase in participation rates of the GSOC project from 80% in 2005 to 89.7% in 2014 stimulates a desire to look into the actions and activities of student developers and to question what attitudes these students have towards code plagiarism.

Despite this growth in open-source codes and related products, as well as the increase in students' involvement in open-source projects, few studies have investigated students' views on academic honesty with respect to submitting open-source codes for class assignments. The primary motivation for this study therefore, is to investigate how technology students relate to coding assignments and what ethical views they hold, regarding code-plagiarism – an "Academic Crime".

LITERATURE REVIEW

The title of the study dictates the broad areas within which scholarly literature can make significant contributions to understanding students' attitude towards code plagiarism. The literature review is divided into two main sections: code-plagiarism and open source software development. This is to deepen our understanding of code-plagiarism and to look at how software development thrives within an open-source approach to computer programming.

CODE-PLAGIARISM

Plagiarism in general involves passing off another persons' work or ideas without giving credit to the author or claiming credit for another persons' work or ideas. Park (2003) gives a contextual presentation of Plagiarism. His presentation covers a definition, an

etymology of the word and some rhetoric regarding plagiarism that have been used in literature. He remarked that although plagiarism is considered today as a vice, in the past, when imitation was considered as the highest form of flattery, plagiarism was seen as a virtue. Code-plagiarism or source code plagiarism as it is specifically called, has no commonly agreed description (Cosma and Joy, 2006). Hage et al. (2010) define it as "trying to pass off (parts of) source code written by someone else as one's own (i.e., without indicating which parts are copied from which author)". However, according to Cosma and Joy (2006), source code plagiarism is not limited to unacknowledged presentation of source codes but includes the copying of structural and lexical organization of source codes. Within such context, detecting source-code plagiarism becomes practically impossible for humans, especially for teachers. An automated system to handle the process is obviously very appealing.

Hage et al. (2010) perform a comparison of five plagiarism detection softwares and conclude that despite the acceptable success of plagiarism detection softwares, further work is required to cover source codes that are written in other programming languages and for varied real life scenarios. However, as Hattingh et al. (2013) remark, such reactive methods have not been complementary with proactive approaches of educating students on plagiarism; defining clear policies and adopting honor codes. This imbalance fosters a "police-bandit" relationship among teachers and students, particularly when it comes to grading assignment and it poisons the learning environment and stifles innovation.

Some researchers (Tomazin and Gradisar, 2006; Paumier, 2009), however, belief that software in academia, both used and developed, should be open source. Paumier (2009) highlights that by keeping software in academia open-source, the software will benefit from collective responsibility and efficiency; authenticity; compatibility with different systems; peer review; continuous research and development; and cumulative work. Within such a framework, the meaning of code-plagiarism becomes questionable and despite the variety of perceptions on plagiarism among faculty (Sutherland-Smith, 2005), the need to rethink the concept of plagiarism becomes eminent.

OPEN-SOURCE SOFTWARE DEVELOPMENT

In almost every industry, a comparison can be made between open source and close source practices. Mattos (2012), talks about the aviation industry and how the openness to information exchange has been the backbone for the success of the industry. In the software industry, the spread of the open source philosophy is closely linked to the Internet boom. With increasing number of users on the Internet, more and more people write and share source codes for various computer software. Consequently, some users see this as an opportunity to get free code and thus the establishment of the historical two camps: the open-source software (OSS) and the free software camps. The Latter is perpetuated by the free software foundation and Richard Stallman while Eric Raymond and various open source initiatives are the main proponents of OSS.

According to Eric Raymond, open-source actions and activities should be profitable. In his article, titled The Cathedral and the Bazaar, Eric Raymond (1999) draws nineteen lessons from observing the way in which open-source and commercial software were developed. These lessons inspire various business models that are exploited to make open-source actions and activities economically profitable. Gomulkiewicz (2004) acknowledges and credits Eric Raymond for taking-on a leadership role in changing the "anti-commercial" reputation of open source projects and expounded on different business models which open-source are exploiting. These models include those used by technology startup companies, top software development companies and some technology giants. Companies, such as Apple and Google, combine different models to maintain vibrancy in the technology market.

Despite the advantage of readily available technical support which proprietary software has over open-source software, their high cost and low continual development (Singh, 2013) provides opportunities for OSS to thrive in the market. Looking beyond national and international markets, Kumar and Singh (2009) explain the key points in national open source software policies and are very particular about the strategic economic benefits which such policies offer. They cross examined the national open source software policies of South Africa, United Kingdom, Denmark, Brazil, Venezuela, Peru and India and concluded that the benefits of a national open source software policy can manifest only if the policy truly captures the spirit of open source software. Their advice to developing countries is to adopt a national open source software policy since it will help develop a local software industry, discourage software importation, and encourage national security.

The successes of open-source software development in local and international markets and even when adopted as a national strategy prick the mind to re-examine its fundamentals and ideologies. Kumar & Singh (2009) and Paumier (2009) reiterate a set of four constrains which the free software foundation poses on software: (1) The freedom to run the program, for any purpose; (2) The freedom to study how the program works, and adapt it to your needs – access to the source code is a precondition for this; (3) The freedom to redistribute copies and (4) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits – access to the source code is a precondition for this. Such constrains on software ensures that the source code remain open forever. Such stands have an obvious different view on plagiarism, particularly on code-plagiarism.

RESEARCH QUESTIONS

In tandem with other researchers, this research seeks an exploratory understanding of code-plagiarism phenomenon from students' perspective. The "learner-centered" approach to higher education as well as the wide spread publication of source codes on the Internet have guided the research to the following five research questions:

1. How do students carry out research for assignments that require them to write computer codes?
2. How do students acknowledge authors of code that are helpful to their research?
3. What are the various positions of teachers with respect to code-plagiarism as perceived by students?
4. How has the possibility of being able to find computer codes on the Internet helped students?
5. What justifications do students advance for engaging in code-plagiarism?

METHOD

This paper uses a constructionist case study approach to qualitative research and explores the relationship between open source philosophy and attitude towards code-plagiarism amongst technology students. Past and current students of two faculties at a public university in Sub Saharan Africa participated in the research. The purposeful sampling of these students was driven by their exposure to open source philosophy and computer software development. Primary data was collected via email responses from the students while secondary data was collected from university manuals, faculty rules and regulation documents and observations. The subsections below describe in details the research participants, the research procedures and data collection.

PARTICIPANTS

The participants of this study were nine students, aspiring either for a minor degree in computer science or a degree in software engineering. They have been referenced in this paper based on how they responded to the initial email. The first respondent (R1) is an ex-student, who graduated with a Bachelor of Science in Mathematics and a minor in Computer Science and participated in both computer programming and software engineering courses. The second respondent (R2) is a senior year student from the faculty of engineering and Technology, who works as a lead software engineer in a local start-up company and is a founder of a start-up company as well. The third respondent (R3) is a senior year student from the Faculty of Engineering and Technology specializing in software engineering. Respondent four (R4) is a sophomore student from the faculty of Engineering and Technology, who has not yet participated in the software engineering course. Respondent five (R5) is a senior year student from the faculty of engineering and technology, who works as software engineer and mobile application developer in a local software company. The sixth respondent (R6) is a Junior year student from the Faculty of Engineering and Technology, specializing in software engineering while respondent seven (R7) is an ex-student who now works as a web master and blogger. The eighth respondent (R8) is an ex-student from the faculty of science, who graduated with a Bachelor in Physics and a minor in Computer Science and now works abroad while the ninth respondent (R9) is a freshman in the faculty of Engineering and Technology.

PROCEDURES AND DATA COLLECTION

An email was drafted, vetted with other colleagues and sent to students who had participated in computer programming related courses in the last five years (see Appendix A). The purpose of the mails was to solicit the students' views and opinion about how they got along with their programming assignments during the course. The mail also required the students to expound on their teachers' impressions about their coding capabilities and code plagiarism. Prior to sending out the email, ethical clearance was obtained from the Research Ethics Committee of the institution where the research was carried out.

An embedded analysis (Yin, 2003) of the response from nine out of ten students was carried out immediately their replies were received. The rejected case was due to the fact that the student did not answer any of the questions, but wrote a one sentence appreciation of his teachers. Strauss and Corbin (1998) approach of line-by-line analysis was implored to generate insight into the data that was collected.

DATA AND FINDINGS

The data has been tallied around the five research questions stated above, based on their appropriateness to the question and also to facilitate interpretation.

RQ1 *How do students carry out research for assignments that require them to write computer codes?*

Some students are very analytical and methodical in the way they carry out research for assignments that requires them to write computer codes. R9 says, "Firstly I will write down the objectives of my research spelling out clearly the PURPOSE of the research … ". However, the students generally acknowledged using the Internet, certain websites, computer books and asking from friends and others. R1 says "I use the internet, computer books to check whether related codes to the research are there; I do also consult friends of the same academic hierarchy or higher that can help me with some suggestions to my research.". Students are fond of the Google search engine and other forums. R3 says " … If I need to learn it to do a project immediately, I use tutorials such as tutorialspoint,

stackoverflow, for rapid answers … Google is my best friend at these moments, to find out exactly what I want … ". R4 and R8 also talk about stackoverflow and tutorialspoint as web sources from where they can find relevant computer codes.

Other students try solving the problem first, and only turn to other sources for solutions if they cannot do the assignments. On this, R4 says:

> [I] write the code my self. I can't, search for and read any material (not code) related to the assignment. Try to implement knowledge gained. If I can't, approach one or two friends. If they can't, take it to online forums". This position is shared by R2, R5, R6 and R7. R7 for instance says "I try to write it myself for the purpose of learning. In the real world where there [is] pressure on deadline if I find some fragment that does exactly what I wanna do, I won't hesitate to use it".

Textbooks were also reported by two of the respondents (R3 and R6) as a lucrative source from which students find code fragments.

RQ2 *How do students acknowledge authors of code that are helpful to their research?*

The comment from R3 summarizes the views of most of the respondents on this issue. R3 writes:

> Generally on forums such as stackoverflow and many others, if a solution was helpful to you, you upvote the solution as an acknowledgment to the author, so that's how i acknowledge authors whose code i use. For sources like textbooks, I usually don't. From other sources like zipped codes that do specific tasks, the authors' names are usually written at the top, for example the top document javadoc section of Java code. I respect their work and keep their names, adding mine below theirs if I did any modifications, or just leaving it as-is if I just used the code.

The view on "up-voting" is shared by R5. Some students would only acknowledge authors whose entire code has been useful to them and where only part of the code was useful, little or no acknowledgement is given.

In general, students acknowledge authorship of helpful codes during presentations of their work (R1) and in the reference and acknowledgment sections of their write-ups (R5, R9) and are particularly appreciative of fascinating codes (R4, R7). On this R4 writes "I acknowledge coders who write code that get me surprised about the simplicity they make a complex assignment look. Coders who write extremely dense code to solve a problem in fewer lines are also acknowledged … ".

RQ3 *What are the various positions of teachers with respect to code-plagiarism as perceived by students?*

Students perceived that the majority of their teachers are against code-plagiarism since copying codes encourages laziness (R4), weakens their programming competence (R5) and makes them dependent. R2 distinguishes between two categories of teachers and writes:

> Some of our teachers have never really been good at writing code and just did so to fulfill righteousness, it would be normal that they sometimes stand for the fact that we just copy and give them as is so long as the problem was solved. The stand is different with some of our motivated teachers. They don't want to get that we copy codes from external sources and use without reference or effort to improve what we get. This is because they believe that just copying and using code makes one lazy and not able

> to produce quality code and always being dependent. They are not the properties of any programmer who wishes to make a good living of his art.

According to respondent six, "The Course Instructors ... so far have not taken any clear position on the copying of code. A number of them have said, though, that one does not learn much when they just copy a solution without putting it in their own terms ". Whereas, for respondent nine, "Some of my teachers encourage [students] to copy codes fragments reason being that it saves programming time … [but the] … Majority say copying codes impedes reasoning and encourages laziness". Respondents eight clearly did not understand this question and writes "I don't understand this point".

RQ4 *How has the possibility of being able to find computer codes on the Internet helped students?*

Most of the students have warned that finding code on the Internet is generally not a straight forward task and could even be more confusing. On this R2 writes:

> … being able to find code made it more or less faster for me to build some modules and implement functionality faster but not necessarily better. Sometimes the times to research code especially for optimal solutions could be longer than if i had to develop one myself ". R5 is critical about code on the Internet and so does R8, who notes by saying " It should be worth noting that copying computer code from the internet is not that easy. First, you must be able to understand the code and make it suitable for your application. Secondly, you might slow that your system by running copied codes that you do not actually need …

On a general note, students are very recognizant and appreciative of the availability of software codes on the Internet. All the respondents say being able to fine computer codes on the Internet has been helpful to them, particularly to save time when they are stocked. R3 writes "Finding code on the internet has helped me many a times, when I got stuck while coding ". Internet code has improved on their efficiency in carrying out programming tasks. R5 comments that

> Its has being of great help, since i have found better (and more efficient) ways of doing some tasks. It has helped me to save time and effort in doing some tasks and i have also learn a lot.". It has also provided them with a platform for peer-evaluation and peer review to boost their confidence in what they are doing. R6 writes "Reading open-source code allows me to see how different persons approach a problem. The way they break down the problem, the algorithms they develop and the various methods they employ to implement these algorithms. It helps me simplify my own approach to problem-solving.

RQ5 *What justifications do students advance for engaging in code-plagiarism?*

Most students actually did have a way of acknowledging authors, whose codes have been helpful to them and did not try to justify why they would not acknowledge them. However, some students (R6) do not deem it necessary to have a reference section for class assignments as they feel that it conveys a different message to their teachers. "As concerns class assignments, I don't deem it helpful to add references to authors when it concerns code. I feel the teacher may get the wrong idea that I just copied the whole piece of code".

Sometimes, students engage in code-plagiarism because of the complications in the acknowledgment process. R8 on this writes:

> … Sometimes, I simply do not acknowledge any authors because of the requirement necessary to acknowledge them. It takes a lot of time to create and account and sometimes I find it not worth creating an account just to acknowledge someone that I will never come back to his/her website. This happens especially when I randomly search for answers on google.

Respondent seven for instance says, "I stand on the idea of don't reinvent the wheel" while R1 puts it this way " … nowadays, it is difficult to innovate in many fields since many 'basic' aspect have been discovered … ". Time management was the burning issue which almost all the respondents highlighted to justify why students copy codes from the Internet.

DISCUSSIONS AND IMPLICATIONS

Plagiarism is an ethical issue. Defining licensing agreements and laws to govern it could be one way of addressing the issue. In universities, plagiarism is considered a high form of academic dishonesty and attracts different sanctions ranging from penalties to expulsion. The discussions and implications of this paper are presented in two folds. Firstly, on students' engagement in code plagiarism, from the perspective of what the open-source philosophy offers and, in line with the research data. Secondly, on the demands which the actions of today's students are placing on University rules and regulations.

STUDENTS' ENGAGEMENT IN CODE PLAGIARISM

The findings presented above show that students are particularly concerned about the amount of time alloted to them for class assignments that require them to write computer programs. They are equally concerned about who is reviewing or correcting the assignment and what grade they get for the assignment. In this light, the discussion and implication of students' engagement in code-plagiarism is presented under the following four sub-themes.

Time Management

Today's student life is characterized by too many activities. For most university students, it is the first time to live out of the hospices of their parents and to be the sole managers of their socioeconomic activities. Furthermore, academic activities and particularly, developing computer software generally requires long hours of work. The student therefore finds him/herself in an arena where being able to manage allotted time could be a cherished virtue but might not always be successful. Cutting corners becomes the obvious option, particularly within the academic milieu, which is perceived as a lesser life-threatening area by some students. Nonetheless, like most of the respondents mentioned, being able to find code on the Internet is a time saver. It gives students the opportunity to focus on the task at hand, deepening their understanding of the problem as they explore solutions of their peers and seniors.

Students' Attitudes towards Teachers and Class

The relationship between teacher and students is very critical to student's attitude towards code plagiarism. Some teachers demand codes in class assignments in fulfillment of the demands of the course. The students' reaction to this is to present a working code and get the teacher's approval rather than on how this code is developed. On the other hand, motivated teachers will always like to see students doing the right things, the right way. Students may or may not develop a receptive attitude towards such teachers but in either case, this defines their attitude towards plagiarism in general and particularly towards code plagiarism. The open source philosophy gives students the opportunity to broaden their

circle of colleagues and friends as they interact with students and teachers who are working on similar and related projects. It rules out the possibility of attaching students' attitude to a particular teacher or classroom setting by providing alternative settings for students to explore.

Efficiency Gain

The desire to attain and maintain a certain academic status puts students under pressure and where circumstances do not permit that these academic levels be honestly achieved, some students resort to unorthodox methods. With respect to code plagiarism, some students capitalize on the inefficiency of the classroom settings. In scenes where a single teacher has to read through millions of lines of code, and with the absence of an automatic code plagiarism detection system, some students shamefully get away with plagiarism. Observations reveal that such are the impractical scenarios in which some software developers are trained. The teachers' incapability to verify whether the codes work or not offers a seductive temptation and even the most brilliant and hardworking students are likely to fall. On the other hand, encouraging students to participate in Open source projects would be ore efficient. The organization and cataloging of code in libraries and projects, which is characteristic of the open source philosophy makes for incremental development and enhancement. In such a setting, the student's contribution is easily visible, genuine and can be verified. This is because, participating in an open source project often requires working for a solution to a reported problem or contributing to a missing piece of the project.

Genuine Lack of Understanding

Sadly, some students unconsciously engage in plagiarism and particularly code plagiarism, out of ignorance. It is the institution's responsibility to state and enforce its policies on plagiarism and to educate the students and teachers on policy issues as well as to define mechanisms that facilitate acknowledgment of authors of codes. An open source approach actually makes this process effortless. The results of this study shows that students readily give credit to authors of codes which they find on the Internet, especially if the code repository has a flexible structure to enhance the process. Within the social network context of open-source projects, code ratings are analogous to article citations.

Park (2003) combines the views of Payne and Nantz (1994), Lim & See (2001) and Evans & Youmans (2000) to show that plagiarism is socially constructed, legitimated and perceived differently by students with different backgrounds. Establishing and supporting this background is largely the role of the educational system as it defines the philosophical and methodological underpinnings that guide and govern what students do in school. The open source philosophy constructs a social context that is governed by praxis. It demands that people (students) should have access to the works of others' and make and share genuine contributions to that work. In academia, this translates into students' authentic contributions as assessed and evaluated by teachers and peers. Within such contexts, code-plagiarism becomes irrelevant since the focus is on scrutinizing, fine-tuning and perfecting codes.

UNIVERSITY RULES AND REGULATIONS

The academic actions and activities of both students and teachers in universities are managed by the rules and regulations; policies and procedures of the given university. In a document defining university standards in Cameroon (MINESUP, 2015), the ministry of higher education (2015) states clearly that it is the university's responsibility to manage researcher and research ethics, through its governance structure. An examination of university policies on plagiarism from the university of Johannesburg (University of

Johannesburg, 2008), the university of South Africa (University of South Africa, 2005) and Harvard university (Harvard University, n.d) reveals that Universities have taken a disciplinarian approach to addressing plagiarism. These documents list out the responsibilities of faculties, academic staff and students and then describe procedures and sanctions relating to alleged plagiarism cases. The implications of such an approach is the establishment of a policed society within academia, where suspicion and skepticism prevails and knowledge sharing is dwarfed.

Benkler and Nissenbaum (2006) define commons-based peer production as a socio-economic system of production that has emerged in the digitally networked environment. The central thesis of their paper is that "socio-technical systems of commons-based peer production offer not only a remarkable medium of production for various kinds of information goods but serve as a context for positive character formation". Their work implicitly links open-source philosophy and character formation in a way that sounds appealing to universities. It is the university's responsibility to build and maintain pedagogic structures that foster virtuous character formation and enhance authenticity of student's academic work. The discussions here point to the fact that adhering to the open-source philosophy promises a possible solution to the moral decadence that drive students into plagiarism and particularly code plagiarism.

THEORETICAL IMPLICATIONS

This qualitative study falls within the realms of using technology in the teaching-learning process. A renown theory in this area is the Technological Pedagogical Content Knowledge (TPACK) framework, extensively reviewed in Chai et al (2013). Three main constituents of the framework are Technological Knowledge (TK), Content Knowledge (CK) and Pedagogical Knowledge (PK). In their systematical review, Chai et al (2013) examined seventy-five (75) published article and found that fifty-five (55) of them collected and analyzed data. They present a summary of these empirical studies (Chai et al, 2013: p36) and classify their themes into ten (10) categories. Strangely, none of these themes addresses ethical concerns of using technology in the teaching-learning process.

The findings from this study show that there is an implicit need to examine ethical issues and concerns of using technology in the teaching-learning process. More so, it shows the need to extend popular theoretical frameworks, particularly TPACK and its derivatives, to incorporate an ethical component. One possibility could be to investigate the relationship between ethics and the other components of TPACK.

RECOMMENDATIONS AND CONCLUSION

The findings of this research and the discussions presented call to mind the consequences of a stereotype mind frame. The maintenance of conventional formulaic university structures conspire only to build a prison for the university community itself. The voices of a representative constituency of technology students echo the need for revision of university structures, especially relating to code plagiarism policies. Although the finding can not be generalized, they shed some light on the issue and suggest the following recommendations and directions for further research.

1. The Delivery of courses that require students to write software codes and generally technology related courses should be approached from a practical perspective. A course should be seen as a set of activities that a student should perform rather than as a body of knowledge that a student should acquire. In this light, it is worth investigating the impact of such an approach on plagiarism and especially on code plagiarism.

2. Under the open source philosophy, the teacher assumes a mentor role, rather than an instructor role and should therefore be an active participant in appropriate open-source projects. Research about the role of teachers in the teaching-learning process is rich, but linking this role to plagiarism and particularly to code plagiarism within different education settings would be worthwhile.
3. Majority of the reasons why students engage in plagiarism is related to the prevailing moral decadence of today's society. It would be worthwhile to examine a Christian approach to managing plagiarism – an approach based on understanding the plight of students.
4. The extension of theoretical frameworks used in studies that investigate of how technology, pedagogy and content integrate in today's classroom. A possible aim would be to incorporate an ethics component.

In conclusion, technology students often use the Internet when faced with exercises that require them to write codes and are therefore inclined to the temptations of code-plagiarism. They use mechanisms found on websites to acknowledge authors of codes and generally have an appreciative attitude towards code on the Internet, especially those that have been helpful to them. However, such mechanisms are seldom understood, appreciated and encouraged by today's educational system. Rethinking the educational philosophy therefore offers a possibility for mitigating students' attitude towards plagiarism and particularly towards code-plagiarism. The open source philosophy, which views knowledge as sacred rather than as secrete; as something to be shared rather than to be secured, offers a commendable starting point.

REFERENCES

Benkler, Y. and Nissenbaum, H. (2006) 'Commons-based peer production and virtue'. *Journal of Political Philosophy*, *14*(4), pp. 394–419. doi: 10.1111/j.1467-9760.2006.00235.x.

Chai, C.-S., Koh, J. H.-L., & Tsai, C.-C. (2013). A review of technological pedagogical content knowledge. *Educational Technology & Society, 16* (2), 31–51.

Cosma G. and Joy M. (2006) *Source-code plagiarism: A UK academic perspective. Research Report No. 422.* Department of Computer Science, The University of Warwick

Davis, S.F., Grover, C.A., Becker, A.H. and McGregor, L.N. (1992) Academic dishonesty: Prevalence, determinants, techniques, and punishments, *Teaching of Psychology*, *19*(1), pp. 16–20. doi: 10.1207/s15328023top1901_3.

Evans F. B. & Youmans M. (2000) ESL writers discuss plagiarism: the social construction of ideologies. *Journal of Education, 182* (3), pp. 49–66.

Gomulkiewicz R. W. (2002) *De-bugging Open Source Software Licensing.* 64 U. PITT . L. R EV . 75, 83 (2002).

Gomulkiewicz, R. W. (2004). Entreprenurial Open Source Software Hackers: MySQL and Its Dual Licensing. *Computer L. Rev. & Tech. J.*, *9*, 203.

Google Developers (2016). *Google summer of codes*. Retrieved from https://developers.google.com/open-source/gsoc/

Goss A. K. (2007). Codifying a commons: Copyright, copyleft, and the creative commons project, 82Chi.-Kent. L. Rev.963 (2007). Available at: http://scholarship.kentlaw.iit.edu/cklawreview/vol82/iss2/24

Hage, J., Rademaker, P., & van Vugt, N. (2010). A comparison of plagiarism detection tools. *Utrecht University. Utrecht, The Netherlands*, 28.

Harvard University. (n.d) Harvard guide to using sources: Harvard plagiarism policy. Retrieved from http://usingsources.fas.harvard.edu/icb/icb.do?keyword =k70847&pageid=icb.page355322

Hattingh, F., Buitendag, A. A., & Van Der Walt, J. S. (2013). Presenting an alternative source code plagiarism detection framework for improving the teaching and learning of programming. *Journal of Information Technology Education*, *12*, 45-58.

Joy, M., Cosma, G., Yau, J.Y.-K. and Sinclair, J. (2011) Source code plagiarism—A student perspective. *IEEE Transactions on Education*, 54(1), pp. 125–132. doi: 10.1109/te.2010.2046664.

Kumar, S. A., & SINGH, P. (2009). Open Source Policy. In *Proceedings of the International Conference of Academia Libraries (ICAL) 2009-Technology, Policy and Innovation*.

LaFontaine, A. (2005). Adventures in Software Licensing: SCO v. IBM and the Future of the Open Source Model. *J. on Telecomm. & High Tech. L.*, *4*, 449.

Lim, V.K.G. and See, S.K.B. (2001). Attitudes toward, and intentions to report, academic cheating among students in Singapore. *Ethics & Behavior*, *11*(3), pp. 261–274. doi: 10.1207/s15327019eb1103_5.

Mattos, B.S. de (2012) 'Open source philosophy and the dawn of aviation', *Journal of Aerospace Technology and Management*, 4(3), pp. 355–380. doi: 10.5028/jatm.2012.04030812.

MINESUP. (January 2015). University standards: Applicable to all higher education institutions in Cameroon. Retrieved from http://www.minesup.gov.cm/ UNIVERSITY%20STANDARD.pdf

Park, C. (2003) In other (people's) words: Plagiarism by university students--literature and lessons. *Assessment & Evaluation in Higher Education*, *28*(5), pp. 471–488. doi: 10.1080/02602930301677.

Paumier, S. (2009). Why academic software should be Open Source. *INFOtheca: Journal of Information and Library Science*, *10*(1-2), 51-54.

Payne, S.L. and Nantz, K.S. (1994) 'Social accounts and metaphors about cheating', *College Teaching*, 42(3), pp. 90–96. doi: 10.1080/87567555.1994.9926831.

Raymond, E. (1999). The cathedral and the bazaar. *Philosophy & Technology*, *12*(3), 23.

Shimel, A. (2012). 10 most successful open source projects of 2012. Retrieved from http://www.itworld.com/article/2827587/enterprise-software/10-most-successful-open-source-projects-of-2012.html

Singh, V. (2013) 'Challenges of open source ILS adoption', *Proceedings of the American Society for Information Science and Technology*, *50*(1), pp. 1–4. doi: 10.1002/meet.14505001115.

Stevens, G.E. and Stevens, F.W. (1987) 'Ethical inclinations of tomorrow's managers revisited: How and why students cheat', *Journal of Education for Business*, *63*(1), pp. 24–29. doi: 10.1080/08832323.1987.10117269.

Strauss, A., &Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Thousand Oaks, CA: Sage.

Sutherland-Smith, W. (2005) 'Pandora's box: Academic perceptions of student plagiarism in writing', *Journal of English for Academic Purposes*, *4*(1), pp. 83–95. doi: 10.1016/j.jeap.2004.07.007.

Tomazin M. and Gradisar M. (2007) Open source software in slovenian primary and secondary schools. *Informatics in Education*, 2007, *6(*2), 443–454.

University of Johannesburg. (17 July 2008). Policy: Plagiarism. Retrieved from http://www.uj.ac.za/EN/Faculties/science/departments/zoology/research/Documents/Plagiarism%20Policy.pdf

University of South Africa. (2005). Policy for copyright infringement and plagiarism. Retrieved from http://www.unisa.ac.za/contents/colleges/col_grad_studies/docs/Policy_copyright_infringement_plagiarism_16November2005.pdf

Yin, R. K. (2003). *Case study research: Design and methods* (3rd ed.). Thousand Oaks, CA: Sage

APPENDIX

THE EMAIL FOR PRIMARY DATA COLLECTION

Dear Students and exStudents,

Accept July greetings and Happy Long Holidays!!
I write following a research I am conducting to solicit your participation in the research. In this light, I will require you to provide answers to the questions below. There is no correct answer and I beg you to write as much as possible for each question (at least 5 sentences). Originality of your responds is most cherished.

1. How do you carry out research for assignments that require you to write computer codes?
2. How do you acknowledge authors of code that are helpful to your research? and if you do not acknowledge them, why?
3. In your opinion, what are the various positions of your teachers about copying computer codes and why do you think they take these positions?
4. How has the possibility of being able to find computer codes on the Internet helped you?

Thank you in advance for taking time off to respond to these questions and Enjoy your holidays

Best wishes.

PS
Please send your reply to me directly and feel free to provide any additional information such as: Level, Institution, Batch, department, other relevant interest, etc ... so that it will make it easier for me to group and classify the responses.