I.J. Modern Education and Computer Science, **2015, 12, 42-50**
Published Online December 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.12.06

An Investigation of Software Engineering Knowledge of Undergraduate Students

Isong Bassey
North-West University, Department of Computer Sciences, Mmabatho, South Africa
Email: bassey.isong@nwu.ac.za

Dominic Afuro and Mbodila Munienge
University of Venda, Computer Science Department, Thohoyandou, South Africa
Email: domafuro@gmail.com, munienge.mbodila@univen.ac.za

Abstract—Computer programming (CP) course offered in universities is difficult coupled with insufficient infrastructures and teaching staff. In spite of these, several undergraduate Computer Science (CS) students are increasingly acquiring programming skills and developing commercial applications even without attending formal programming classes. However, software intended for use other than by the developer requires teamwork, the use of software engineering methodologies and quality. What is not known about these undergraduate students is how their programming is learnt or applications developed. This is important in the light of software dependability and cost of failures today. Therefore, this paper investigates how undergraduate CS students learn programming and their software engineering knowledge. The purpose is to gain insights into how knowledge is gained and applied. To accomplish this, the paper conducted a survey utilizing questionnaire and interview on undergraduate students of CS in the University of Venda (UNIVEN). The data collected were analyzed and results quantitatively and qualitatively presented. The results showed that many CS students learned programming via the Internet reusable code, applied development methodology and are aware of software quality during development.

Index Terms—Computer programming, developer, teaching, learning, students.

I. Introduction

Today, in the context of continuing pressure for well-trained personnel for economic development with respect to information technology, Computer Science (CS) graduates have been in high demand [12]. In this case, programming skill is one of the core competencies CS graduates are required to have. For this to be achieved in the perspective of higher education, computer programming (CP) courses are offered. CP courses are intended to prepare and proffer undergraduate CS students for a career: not only with technical knowledge, but also with the skills that are essential to work in real-life software development projects [1]. However, learning to program is hard and students have found programming to be difficult [2][3].

Interestingly today, in the light of the difficulties associated with CP, majority of CS students can still develop software codes exceptionally even without attending formal classes or practical in the computer laboratory. The software applications developed by these students range from personal to commercial. Nonetheless, in the context of commercial applications, it is intended for use by someone other than the developer. Thus, the development of such applications requires teamwork rather than individuals using appropriate software engineering (SE) methodology. The application of SE methodologies is required to develop a high quality system that is reliable, stable, and maintainable throughout its lifetime [4]. It is designed to support professional software development that ranges from the techniques of program specification to maintenance, none of which are considered relevant for personal software development [4].

In addition, as software is becoming more dependable and pervasive day by day, the relevance of software quality cannot be overemphasized. The increasing prominence of software and the related cost associated with software failures are the drivers for high quality of software products. To this end, SE methodologies are indispensable requirements for effective software development in order to meeting such demand. In software development organizations, software processes are used to achieve the required levels of productivity and quality. Thus, CS undergraduate students as future developers can as well follow such processes if their goal is to achieve high quality products in their career. Albeit, it is hard to rebuild the real-world software development in the classroom, student developers need to have skills such as analytical thinking, creative synthesis, and attention to details in order to write good code, test and maintain software to ensure that the developed application meets the needs of the users [5][6].

Though undergraduate CS students are increasingly developing software applications today, commercial applications in particular, the issue is how are such applications developed: *Are SE methodologies employed? Do they develop with the goal of software quality in mind?*

Therefore, the objective of this paper is to explore and answer the questions stated above. The goal is to gain insights into how student applied the knowledge acquired in their studies in their personal software development life and if there are appropriate. This is important in order to provide them with the necessary assistance or suggests some improvement in the current CP teaching methodology that will align to their needs. To achieve this, this study carried out a survey on undergraduate students of the department of CS in the University of Venda (UNIVEN). Reponses obtained were analyzed and results presented quantitatively and qualitatively. The findings indicated that reusable code on the Internet constitutes the new method for learning programming among the undergraduate CS students. In addition, it is also revealed that some of these student developers have software quality in mind when they develop their applications as well as the use of software development methodologies. However, regrettably the students were found not to be following the appropriate methodology and there are not working in teams, which we believed could affect the quality of the developed products.

The rest of the paper is organized as follows: Section II is the study background information, III is the challenges of CP teaching in UNIVEN and IV is the methodology used. Accordingly, Section V is the results and discussions, VII is the threats to validity and VIII is the study conclusions.

II. Background Information

CP is a crucial part of computing and a useful skill which constitutes one of the nucleus competences expected of anyone in the discipline of CS [1]. It is described by Deek & McHugh [7] as a problem solving process of formulating, planning the solution, designing the solution, translation, testing, and delivery. Furthermore, in order to achieve these processes, [7] stressed that programmers must be equipped with skills such as learning the language, composing new programs, comprehending, reusing and integrating existing programs, modifying etc. Nevertheless, programming is difficult and constitute a challenging tasks which involve huge cognitive activities [1][5][7][8].

Today, it has been recognized that one of the greatest concerns that have remained for decades is the teaching and learning of programming [2][3][8]. The teaching and learning of programming has been one of the key challenges in the field of CS. Learning to program is known to be difficult due to the fact that the learning process is vulnerable to several risks [8]. This is evidenced in several researches in the literature [6]. However, these challenges are common in several developing nations' universities. It has been noted that some of the factors that contributes to the challenges are lack of experienced programming teachers, lack of computing infrastructure and ineffective teaching methods [6]. In most cases, CP is only taught in ordinary classrooms while computer laboratories are rarely used. This has resulted in many students that offers CP as a course to lose enthusiasm and interests in the learning [1]. In addition, it has been revealed that most students' interest in the CS discipline is declining. For instance, data by Foster [9] showed that students who indicated their desire to major in CS declined by more than 60 percent between the fall of 2000 and 2004. Nevertheless, in spite of the difficulty associated with CP, several undergraduate students are progressively developing software applications (either personal or commercial) as well as improving their skills day by day. But what has not been known is how the programming is learnt and what software engineering methodology is applied during the development process, in particular the commercial applications. This forms the basis for this study. This is important because, with today's globalization, rapid technological development and knowledge-based economy [12], it is of the essence to ensure that graduates of CS are fully equipped with the cutting-edge technical skills and basic competences to excel in the production of quality software products.

III. Computer Programming Teaching in Univen

As stated above, learning to program has been deemed hard and the undergraduate programming courses are generally considered insufficient and difficult as well. Consequently, utmost dropout rates among CS students have been witnessed and are popular among several rural universities in developing countries, where UNIVEN is no exception. In the department of CS, there are few teaching staffs, few programming modules and a single computer room with almost 15 to 20 computers while about 200 or more students struggle to learn programming in one computer room with only one or no supervisor. With the nature of the computer laboratory, the lecturer responsible usually employed student tutors to take charge of the students. Sometimes, the student tutors are not different from the students they tutored. In this case, students often get stuck in problems and left unattended to. Consequently, several students have quite left CS programme to other programmes or performed poorly because of lack of motivations as they cannot find ideal solutions to their problems. Moreover, another course that is used to supplement CP is the SE modules where students are given projects in teams.

However, one well-known truth supported empirically that involved the teaching of CP which has to be known is the fact that a motivated student needs some form of guidance to succeed irrespective of the prevailing conditions including the teachers [10][11]. Similarly, ill motivated student will fail regardless of what the teacher says or how good the teacher is at explaining the concepts of CP [10]. Thus, students learning motivation and effectiveness can be hindered by environmental factors such as the learning approach, infrastructural availability and social pressure from learning peers [12]. Though undergraduate students in UNIVEN may be affected by lack of full guidance when it comes to learning CP, it is fascinatingly to know that the students never back-off. They have found a new way of complementing their

classroom's programming needs and discovering it therefore, forms the basis for this paper.

IV. Research Design

This section discussed the design of the research we employed in this study: the participants, the questionnaire and interview design.

A. Study Participants

The target participants in this study are the undergraduate students of UNIVEN who are in their 2nd, 3rd and 4th year that have offered or still offering any of the two programming modules and SE module in the CS Department: COM 1721, COM 2701 and COM 3620. COM 1721 is an introduction to object-oriented programming module in C++ that is taught at the first level and COM 2701 is programming module in Java taught at the second year level. In this study, the first year students were excluded because of insufficient programming experience. COM 3620 is a SE module taught at the third year level. In all these courses, lectures are always held in the classrooms, practical laboratory scheduled maybe once or none, and projects are given in teams where problems are solved using their personal computers since the computer laboratory are not sufficient. The study was conducted using a sample of 112 students (students available as of the time the questionnaire was administered) drawn up from the entire CS student population as captured in Table 1.

Table 1. Participants' Gender and Study Level

Gender		***Study Year***	
Males	92	2nd	65
Females	20	3rd	41
Total **112**		4th	6

B. Questionnaire, Interview Design and Data Collection

This study used the mixed research method, utilizing questionnaire and interview as methods for collecting data from the students. Due to the nature of the information collected, the study followed a descriptive analysis approach to analyze the extracted data. Furthermore, closed-ended questionnaire was used and was subjected to strict evaluation by an independent expert to ascertain the suitability of the questions. The validated questionnaire was used to elicits information from the students such as how they learn programming, develop applications, software engineering knowledge and application in relation to software quality. Based on the questionnaire, the analysis was performed question by question as presented in Section V.

In the same vein, the interview focused only on a subset of the third year CS students due to time constraint, albeit we could have obtained more valuable information using only personal interviews in the entire study. However, the interview was based on the semester project of COM3620 given to third year students in teams of five students each. The project was about developing a medium-size application of their choice using any software development methodology. After the project completion, some selected team members (about 10) were interviewed. The questions were specifically on the application type they developed, technologies used and their teamwork experience. A transcript of this interview was documented and discussed qualitatively in this paper.

V. Results and Discussions

This section presents the summary of the results obtained from the study based on the data collected. The data is analyzed and the results obtained quantitatively presented section by section as stipulated in the questionnaire design.

A. How Students Learn Programming

To explore how students learn their programming, several background questions were asked to obtain the needed information from the students. These questions are as follows.

Do you program?: The students were asked this question in order to improve the quality of the study by focusing on those who can program. This was necessary because CP has been deemed difficult which led to the belive that not all students can program. However, as presented in Fig. 1, the analysis shows that majority of the students, about 93% (104 students) are developers while only 7% (8 students) are not developers.

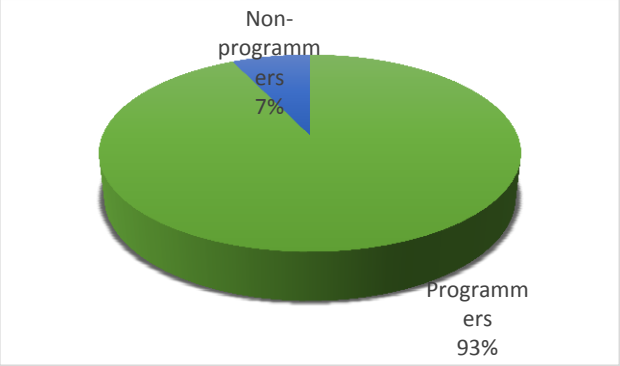

Fig.1. Programming Interest

Moreover, when asked the programmers how long they have been programming, the analysis shows that majority of the students, about 55% have been programming for over a period of one year followed by 31% who have been programming between 2 to 3 years. (See Fig. 1) therefore, based on these results, it can be deduced that despite the difficulties and other environmental challenges students in UNIVEN might face in the course of learning of CP, they have continued to show great interest in CP and their expertise is growing.

What is your general perception about programming?: This question was designed to test students' perception about programming in general. However, from the results obtained and out of the 104 students who can program,

about 55 students considered CP to be interesting, 45 considerd it difficult while 4 declined. This is captuured in Fig. 2. Further analysis revealed that about 64% (66) of the students weren known to have average basic programming skill. Based on this result obtained, this study believed that the result is in line with the response obtained in their years of experience as it shows that the students are gradually becoming proficient in CP course.

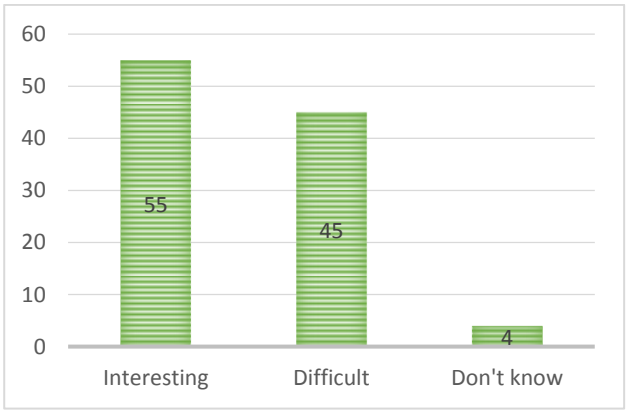

Fig.2. Participants' Programming Perception

What resources are used when learing to program?: This question forms one of the core objectives of this research. It was designed to discover the different approaches students invented in learning CP. Nonetheless, with the analysis presented in Fig. 3, out of 104 students who can program, analysis shows that majority of the students, about 49 of them learnt from reuse codes on the Internet, while 32 learn how to program by combining classrooms, textbooks and reuse codes. The results obtained in this regard suggest that while programming is insufficient in the classrooms, the Internet has become a leverage. Moreover, this could be the reason why students find programming interesting because of the availability of source codes on the internet and textbooks.

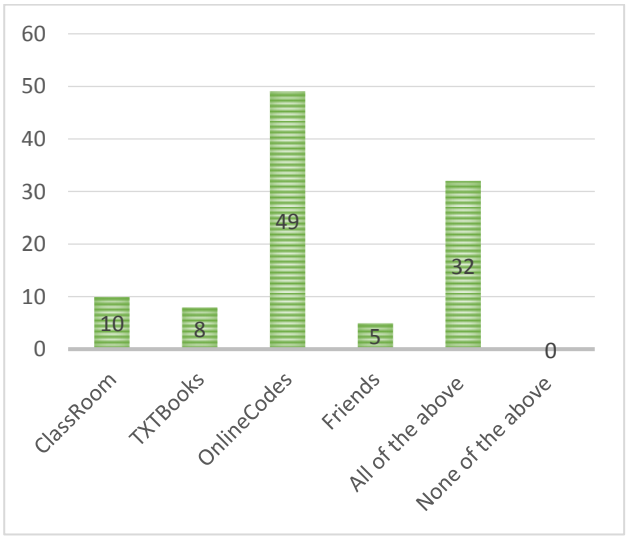

Fig.3. Programming Learning Approach

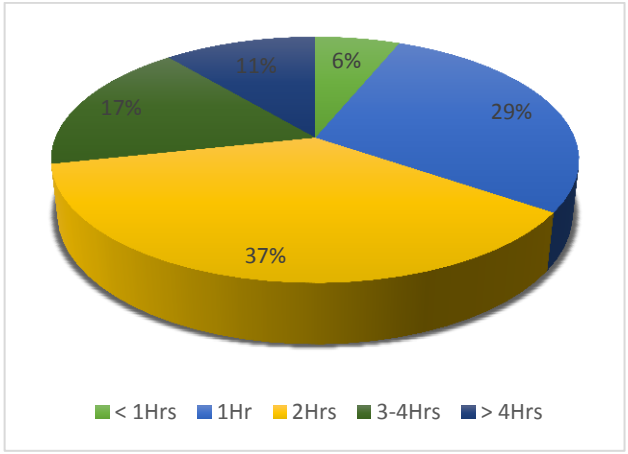

Fig.4a. No of Hours Spent Online Learning Programming

If you have learnt programming with reuse codes online, how often do you access the Internet?: This question was designed to assist this study to know the length of time and how often students used the Internet to learn CP with reuse codes for them to be effective in programming. Interestingly, out of the 81 respondent who uses Internet, analysis indicates that about 29% (23) of the students spent one hour while 37% (30) spent two hours on the Internet learning with reuse codes. (See Fig. 4a) In the same vein, further analysis indicates that 33% (27) of the students accessed the internet once a week, 43% (35 students) access the Internet more than once a week. This is captured in Fig.4b. The significance of this result is that, for students to be effective in programming outside the classroom, they have to spend a considerable number of hours at least more than once a week on the Internet either at their home, school, café and so on.

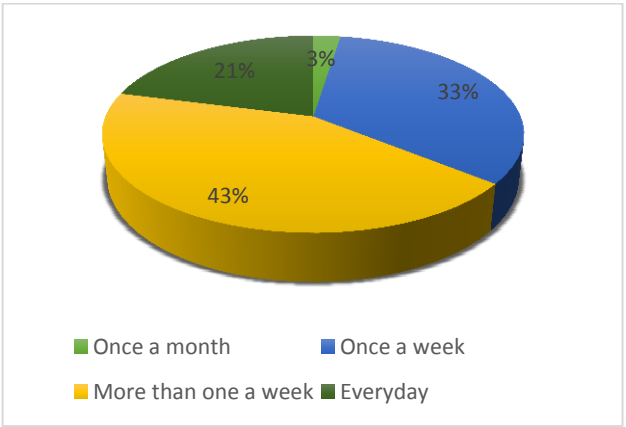

Fig.4b. Frequency of Internet Access

B. Rationale for Learning Programming Outside the Classrooms

This section focus on eliciting information on the challenges students faced and their motivation to learning programming in a specific way. Based on the responses analyzed above, we focused only on the classroom as follows:

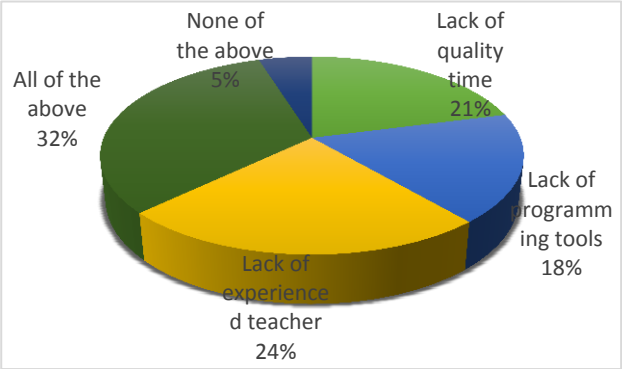

Fig.5. Programming Challenges in Classroom

What is/are the reason(s) why you don't learn programming in the classroom?: This question was specifically chanelled towards obtaining information from the respondents who learnt programming using other methods other than in the classrooms. The justification is to gain insights into what the problems were. However, the analysis presented in Fig. 5 shows that out of 62 respondents, about 32% (20) of students admitted lack of quality time to teach and explain programming concepts in the classroom, lack of programming infrastructures (e.g. well-equipped computer laboratories) and experienced programming teachers. The result obtained here is in line with the study of [6] and by implication, this could be the reason why students invented their own ways of learning CP other than in the formal classroom.

If "lack of quality time" in the classroom is your reasons why you don't learn programming formally, how often are programming classes scheduled in your institution?: Due to the fact that defficiencies in classroom studies have been considered as some of the reasons why students have devised their own approach of CP learning, this study went further to explore how time and frequency of CP classes affects them. However, the analysis obtained indicates that out of the 13 responses that admitted is lack of quality time, about 54% (7) of the students went further to admit that CP class is only scheduled once a week while 46% (6) said CP classes are scheduled not more than twice a week. (See Fig. 6a)

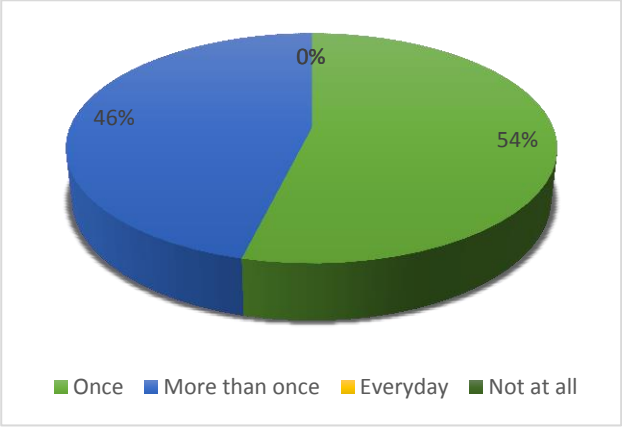

Fig.6a. No Programming Classes in Each Class

Further analysis indicates that 32% of the students spent only one hour a day in each CP class while about 21% spent 3 to 4 hours in a class per day. (See Fig. 6b) With this result, it is so encouraging to see that the frequency of CP classes and the number of hours spent in each class is not sufficient for students to acquire the needed programming skills. Based on this result, the above fact can be said to be one of the reason some students have adopted their own approach through reused codes.

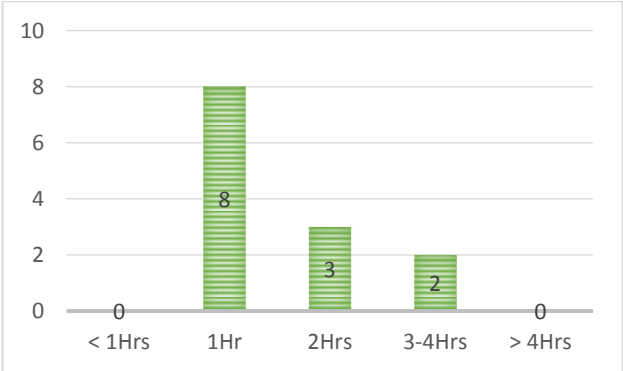

Fig.6b. No Hours Spent in Each Class

C. Software Development Activities

The questions in this section were designed to target only student developers who have developed software applications for personal or commercial use. The rationale is to help the study understand their level of awareness as regards to professional development with respect to software quality. The following questions were then asked:

Have you ever developed a complete working software application?: This question was geared towards knowing how many students were actually making progess in programming and have developed atleast a complete workable system. However, with the analysis, of whom 104 respondents are programmers, about 52% (53) admitted they have developed workable applications, 48% (49) have not while 2 students declined. Additionally, for those who have developed complete software applications, analysis indicates that majority of the students, about 60% (32) have developed complete systems for personal use such as course assignments, course projects, and so on, while only 40% (21) have developed software for their clients based on monetary contract.(See Fig. 7)

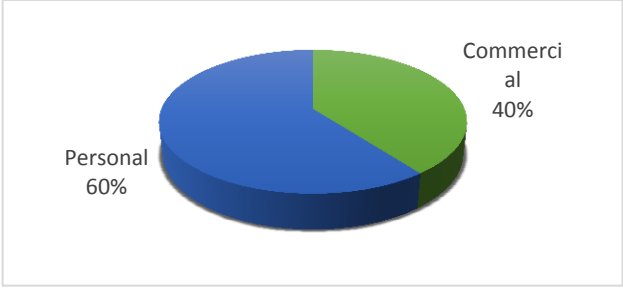

Fig.7. Type of System Students Develop

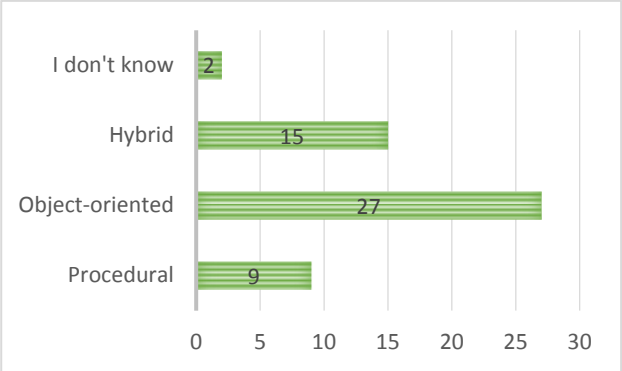

Fig.8a. Software Development Technique

If you have ever developed a commercial software system, how do you go about developing them?: This question was designed to discover if students work individually or in collaborations with others during the course of development. The essence is that today, the development of software applications for market purposes requires joint efforts of members of one or more teams and not a single person [13]. Unfortunately, the results obtained shows that only 32% (17 students) worked in teams while about 68% (36 students) worked individually. Moreover, 27 students have adopted the object-oriented development approach while 15 students combined both procedural and object-oriented techniques (See Fig. 8a). Significantly, these results indicates that majority of the students are still not aware of the importance of working in teams which perhaps, can speed up the development process and promote knowledge sharing. The good thing with the result obtained here is that undergraduate CS students are beginning to use modern development techniques such as object-oriented or a combination of both procedural and object-oriented to develop their applications.

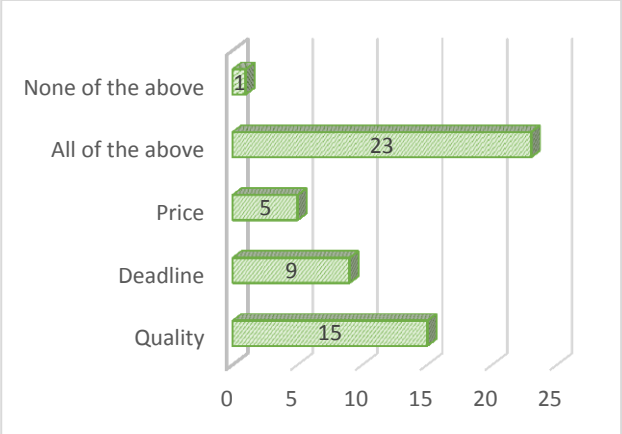

Fig.8b. Software Development Priority

When you develop software, what is your greatest concern?: This question was aimed at assessing what comes to the students' mind when developing software applications in terms of software quality, cash, deadline, and so on. The analysis presented in Fig. 8b shows that, out of the 53 students that can develop full application commercially, only 15 students are actually concerned about software quality while 23 students are concerned about quality, price and deadline. For those with concern about software quality, further analysis has it that out of 38 students, 55% (21 students) believed a quality software is one with the right functionalities, satisfies users' needs and is maintainable. (See Fig.9) Based on the result obtained, we can deduced that some UNIVEN CS students are aware of the importance of software quality, delivery date and perhaps, being paid for their time and effort utilized. Moreover, they are well-informed about software quality but what is not known at this point is if they actually put it into practice which is beyond the scope of this study since this study did not access their developed applications.

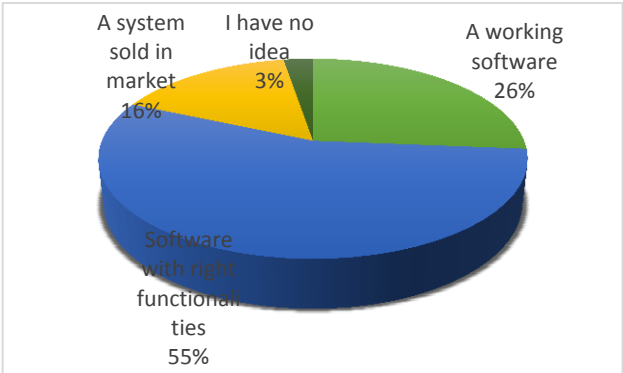

Fig.9. Software Quality Knowledge

Have you ever used software development methodology?: This question was designed to target students who have developed complete software applications. This is important to enable this study assess if they actually applied SE approach when developing their software applications. Nevertheless, the results obtained shows that out of the 53 students, about 75% of students admitted the usage of SE methodology while about 25% have not. Furthermore, analysis based on those who have used SE methodology (40 students) indicates that about 21 students admitted the application of SE methodology is for professional software development.

However, for the students who have used SE methodology, about 40% (16) have adopted the Waterfall approach while 23% (9) students have adopted the Reuse-oriented approach.(See Fig. 10a) To further assess if the students rightly applied the SE methodology they adopted, 24 students admitted they always starts with requirements collections before performing the actual development. (See Fig. 10b) What is interesting in the results obtained here is that several students have used different methodologies and majority of them actually knew the right activity to perform when adopting a particular methodology in order to be on the right track in meeting customer's needs and ensure their satisfaction. However, what cannot be established is the suitability of the methodologies they applied.

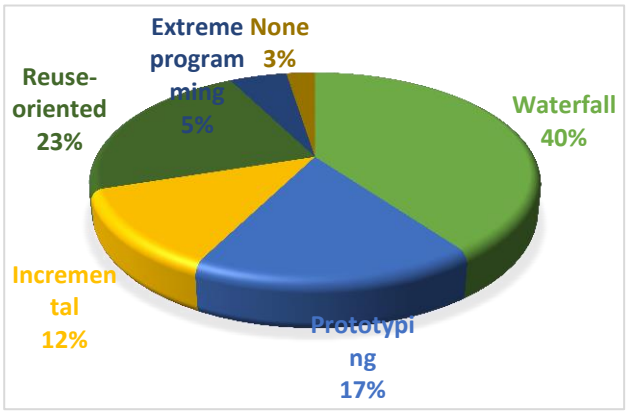

Fig.10a. Software Methodology Adopted and Activities

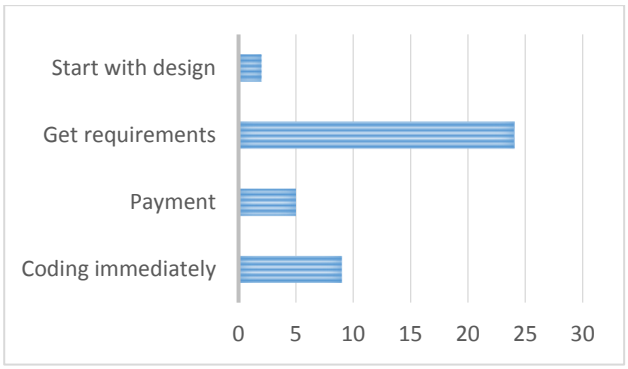

Fig.10b. Software Methodology Adopted and Activities

*If you have used reusable codes, can you align it with any of the methodologies you know?:*This question was designed to explore the appropriateness of the methodology used targeting the students who are using reusable codes from the Internet. However, in an attempt to align their methods with any of the existing methodologies, analysis indicates that out of the 81 students who uses online codes, about 31 students indicated chosed Waterfall model while only 9 students admitted is reuse-oriented approach. Moreover, when asked about their perceptions on the use of online codes, analysis indicates that 58% (47 students) believed using reuse codes makes the work easy and saves time. (See Fig.11)

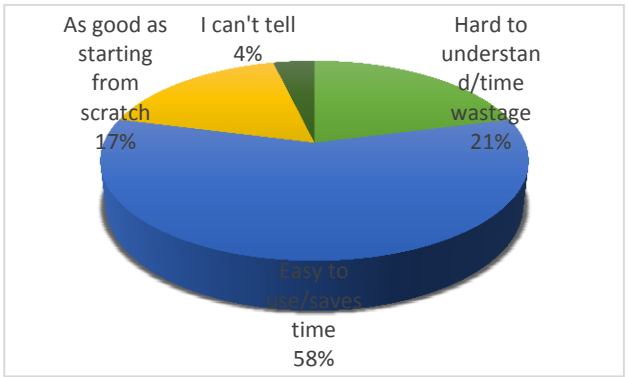

Fig.11. Methodology Alignment and Reuse Codes Benefits

Based on the results obtained here, analysis has revealed that albeit majority of the students used reuse codes, only few students can actually align it to reuse-oriented methodology and as well followed the appropriate steps to develop a quality product. Thus, necessary measures have not been taken to ensure the quality of the product they developed. With this fact, it is believed that using online code is easy and save programming time needed to deliver the software application well as low cost.

D. Interview Transcript

This section discuss the responses documented from the interview conducted on the third year students offering COM3620. The interview focused on two key aspects: programming language/application types and the issues of teamwork. The interview goes as follows:

What are the types of software applications you develop and the languages used?: The goal of this question was specifically to explore the software applications students developed for their clients and the type of programming languages they utilized. Based on the responses obtained, this study found that some UNIVEN CS students specializes in developing two kinds of applications: web-based and desktop based application for their clients. They went further explaining that:

"...for the web-based applications (e.g websites, portals) we used technologies such as PHP 5, JavaScript, HTML 5, CSS, and Mysql, while for desktop applications such point of sales system, we used Java and Mysql only".

When further asked why they have used only these technologies in their development, they responded by saying:

"...those technologies were the only ones we are good at. However, technologies such as C++, C#, visual basic, android, JFX and J2EE, can only be adapted to with the help of reuse codes online whenever our clients demand applications to be built with them".

The transcript represented above is from one of the interviewee, although they all responded the same way. However, from this transcript, it is clear that, although the only programming languages taught at UNIVEN are C++ and Java, some of the students have gone further to learn other languages such as MySql, C#, PHP, CSS, JavaScript, etc. on their own. This confirmed that using online code could been beneficial and productive.

What challenge do you face when developing in teams?: This question was based on the feedback from the project task they were assigned to perform in teams. To answer this question, the students tried to explain the challenges they faced with some of their semester projects in particular, COM360. As explained:

".....in that project, the lecturer allowed us to form teams by ourselves and during the selection of team members, students only selected team members based on their relationships rather than the essential skills/experiences required from each individual for the project success".

Moreover,

"…students have no respect for their fellow students because we are all mates in the same level of study …they found it difficult to show respect for one another. Also, most students don't trust each other when it involves performing a task together".

From these responses, it be seen that most of the team members who have more skills and experience than others do not want to share their knowledge with their colleagues. Instead, they want to work alone and be the only ones known with such skills. They students further stressed that, all these factors affected their projects negatively and consequently, they prefer to develop their application individually other than in teams. Though, this is a good solution for the students, it is highly against the ethics of professional software development. Hence, steps should be taken to address these challenges in future semester projects.

VI. Validity Threats

In this study, there are two threats which could affect the results discussed: internal and external validity threats. Threats to the internal validity of this survey could be as a result of lack of seriousness on the part of some students. During the course of administering the questionnaire, few students were found discussing with friends while responding to the survey. Thus, care must be taken when generalizing the results. In addition, this survey is voluntary and there is the possibility that the results may be biased. The undergraduate students sampled in this study may not represent the actual interest of the entire population of CS students in UNIVEN. In the same vein, a threat to the external validity could be that this study was done at one rural university and the results may not apply to all institutions either in rural or urban areas.

VII. Conclusions

This paper explored software development activities of undergraduate CS students in an attempt to gain insights into how they learn and proficient in programming despite the difficulty and cognitive activities associated with CP. The study also investigated their knowledge of software engineering methodology and their application during the process of development. It was conducted because today, several undergraduate students can develop commercial applications even in the face of insufficient programming courses, teachers, infrastructures, and so on. Furthermore, software quality has been a great concern due to the cost of software failures. Therefore, it is important to know whether knowledge was gained in their studies and if the products students develop are of quality or not. Based on the questionnaire survey (for entire students) and interview (for subset of third year students) carried out, data collected from student programmers were analyzed and results presented quantitatively and qualitatively respectively. From the results obtained, this study found that students learning was effective as most students were able to apply the knowledge acquired to their development activities. Other findings are as follows:

- Several CS students in UNIVEN are developers who have been programming for a year or two with average basic skills and found programming to be interesting.
- Students consider online source codes, in addition to formal programming classes and textbooks as the best approach to learn programming. They spent a significant amount of time on the internet practicing programming with reusable codes.
- Students are discouraged by the time spent in the classroom during programming classes, lack of infrastructures and insufficient programming teachers. Programming classes are schedule mostly once in a week with a maximum of one hour each.
- Several UNIVEN CS developers have developed commercial applications for their clients and are not motivated by cash but to solve real-world problems, add value to the IT industry and to get a good job after graduation.
- Majority of the students enjoyed working individually rather than in teams due to issues of respect, trust, knowledge sharing and so on.
- Students have applied different software development methodologies in their work but unfortunately, majority of them could not align their work to the appropriate methodology, though followed the right activities.
- Majority of the students confirmed the benefits of reuse codes, expressed concern about Internet security but most are not concerned about intellectual property rights as well.

Based on the above stated findings, here are some of the recommendations this study believes should be considered:

- More has to be done to keep the students completely on the right track of preparing and equipping them with the effective and professional programming skills. We believed this will assist them to impact the world of software development. Though students can learn from reuse code on the Internet, real understanding requires learning which is active in the laboratory environment with teacher's guidance for ensuring reflection on the experience obtained from problem solving purposes, otherwise passive programming learning will result in failure.
    - Programming classes schedule in a week and the number of hours spent in a day in the class should be reviewed and revised since they could discourage students from learning programming formally.

- When teaching software development models in SE, more emphasis should be placed on reuse-oriented approach in order to equip students with required skills of using reuse code or components for software quality.

- Course teachers should educate students more on the importance of teamwork/trust and encourage them to practice it so that students who are going to work in the software development firms will not be found wanting. Professional software is developed in teams rather than by individual bodies. In this case, the formation or selection of team members should be the sole responsibility of the teacher and not the students due to issues of trust, respect as well as knowledge sharing.

These recommendations should be taken seriously and be considered to ensure that undergraduate students of CS are equipped with the effective and professional software development skills expected of them to fit into any software firm and excel in their respective workplace after graduation.

REFERENCES

[1] Chris M.Y., Victor C.S. & Lee Y.T. Yu. 2010. Learning motivation in e-learning facilitated computer programming courses. Computers & Education 55, pp. 218–228.

[2] Jenkins, T. (2002). On the difficulty of learning to program. In: 3rd annual conference of LTSN-ICS.

[3] Gomes, A., & Mendes, A. J. (2007). Learning to program – Difficulties and solutions. In International Conference on Engineering Education – ICEE 2007, Coimbra, Portugal.

[4] Sommerville, Ian. 2011. Software Engineering (9th edition). Pearson Education. ISBN-13: 978-0-13-703515-1.

[5] Lam, M. S. W., Chan, E. Y. K., Lee, V. C. S., & Yu, Y. T. 2008. Designing an automatic debugging assistant for improving the learning of computer programming. Lecture Notes in Computer Science, 5169, 359–370.

[6] Robins, A., Rountree,J., Rountree, N. 2003. Learning and teaching programming: A review and discussion. Computer Science Education. Vol. 13, No. 2, pp.137-172.

[7] Deek, F.P., & McHugh, J.2003. Problem Solving and Cognitive Foundations for Program Development: An Integrated Model. Proceedings of the Sixth International Conference on Computer Based Learning in Science (CBLIS), Nicosia, Cyprus, pp. 266- 271.

[8] Lui, A. K., Kwan, R., Poon, M., & Cheung, Y. H. Y.2004. Saving weak programming students: Applying constructivism in a first programming course. SIGCSE Bulletin, 36, pp.72–76.

[9] Foster, A. 2005. Student interest in computer science plummets. Chronicle of Higher Education. http://chronicle.com/free/v51/i38/38a03101.htm.(Accessed on August 31, 2006).

[10] Brito, M.A. and Sá-Soares, F. (2013). Assessment frequency in introductory computer programming disciplines, Computers in Human Behavior, 2013.

[11] Wulf, T. (2005). Constructivist approaches for teaching computer programming. In Proceedings of the 6th conference on information technology education, Newark, NJ, USA: ACM.

[12] Law, K. M. Y., Sandnes, F. E., Jian, H., and Huang, Y (2009). A comparative study of learning motivation among engineering students in South East Asia and beyond. *International Journal of Engineering Education*, 25(1), pp.144–151.

[13] Thomas, P.S., Fernández, R.F. & Manjón, B.F.2009. Learning teamwork skills in university programming courses. Computers & Education 53, pp. 517–531.

Authors' Profiles

Isong Bassey received B.Sc. degree in Computer Science from the University of Calabar, Nigeria in 2004 and M.Sc. degrees in Computer Science and Software Engineering from Blekinge Institute of Technology, Sweden in 2008 and 2010 respectively. Moreover, he received a PhD in Computer Science in the North-West University, Mafikeng Campus, South Africa in 2014. Between 2010 and 2014 he was a faculty member of the University of Venda, South Africa and a Lecturer of Computer Science and Information Systems. Currently, he is a Lecturer in the Department of Computer Sciences, Mafikeng Campus, North-West University. His research interests include Software Engineering, Requirements Engineering, Software Measurement, Maintenance, Information Security, Software Testing, Mobile Computing and Technology in Education.

Dominic Afuro holds B.Sc. (Hons) degrees in Computer Science from the University of Calabar, Nigeria in 2004 and Computer Science and Information Systems from the University of Venda, South Africa in 2014 respectively. He is currently studying for his Masters degree in Computer Science. His research interests include Software Engineering, Mobile Cloud Computing and Web Service Discovery.

Munienge Mbodila holds B.Sc. (Hons) degree in Computer Science at the University of Fort Hare and M.Sc. degree in Computer Science in 2013 at the North-West University, South Africa. Currently, is a PG Diploma in Higher Education student in Teaching and Learning at Stellenbosch University, South Africa. He joined the Department of Computer Science and Information System of the University of Venda as a faculty member and as a Lecturer in 2009. His research interests include Computer Networks, Wireless Sensor Networks, Software Engineering, ICTs and Web Technology in Teaching and Learning.