Global Scientific JOURNALS

GSJ: Volume 8, Issue 12, December 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

EFFECTIVE COMPUTER PROGRAMMING DELIVERY IN TERTIARY INSTITUTIONS: CHALLENGES AND PROSPECTS – A REVIEW

*** Njoku Christian C. ** Oladimeji Adegbola Isaac, ***Yusuf Hussaini Amana**
Computer Science Education Department
School of Science
Aminu Saleh College of Education, Azare, Bauchi State, Nigeria

Abstract - *Considering the advancement in technology in recent times, the knowledge of computer and its modus operandi is no longer an option. Computer programming is the core foundation of computer science education. Learning programming has been a difficult task for students especially those in first year in tertiary institutions, hence the need for teachers to look for methods to make it easier for them to learn. This paper therefore considered the approaches to effective teaching and learning of programming in computer science, it also discussed the challenges to effective teaching of computer programming which include among other, inadequate computer literacy, lack of access to computer and other resources, inadequate experienced teachers in computer programming etc. The paper also looked at some pedagogical methods that can enhance the delivery of computer programming such as Pair programming, Problem-based learning, and puzzle-based learning among others. Finally, recommendations were made for effective teaching of computer programming.*

Keywords: Computer Science, Computer Programming, Teaching and Learning,

I. Introduction

That Computer science has become part of our lives is no longer news as it has influenced and is still influencing the way we live and do things. It is a major enabler for discovery and innovation in most other fields of endeavour. It has variety of fields which could be theoretical or practical. For example, computer graphics which deals with real world visualization of applications and Computational Complexity theory – which deals with explores the fundamental properties of computational and intractable problems and are highly abstract. It has other disciplines such as programming language theory, which considers various approaches to the description of computation, while the study of computer programming itself investigates various aspects of the use of programming language and complex systems.

Computer Science (CS) is a discipline that explores how to use computers to solve real world problems, whose aims include providing fundamental knowledge to the students, developing in the students the skill of thinking and analyzing, create interest and scientific attitudes, make the students associate the knowledge in real time application, among others. Therefore, it is crucial to create an active learning environment to improve students' comprehension and retention of material, allow students to take control and regulate their own learning, and eventually empower them with necessary skills to solve problems outside of the classroom. (Gao and Hargis, 2010).

Computer programming, which as a discipline in computer science is the process of writing instructions that gets executed by computers. The instructions, also known as code, are written in a programming language which the computer can understand and use to perform a task or solve a problem. In computer science (CS) discipline, programming constitutes one of the pillars of computing. It is considered a useful skill which is one of the nucleus competences expected of every of CS graduate (Isong, 2014).

Bebbington (2014) sees computer programming as the process of designing and building an executable computer program for accomplishing a specific computing task. Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as coding). Hence, computer programming can be seen as a method of instructing the computer to do work

and in a specified way or manner. The first computer program is dated to 1843, by Lady Ada Lovelace when she published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.

Program source codes can be written in any programming language of choice. The rationale behind programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem. The process of programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic. It is hard to find any technological gadget today which works without computer programs. Hence, the need for its proper teaching and learning.

The major purpose of teaching computer science is to enable students to grasp the basic knowledge needed for further study of computer science and the related technology and to understand its application. Secondly, it assists acquire the skills of practicality, develop the capacity to think further and apply those skills in real life situations. No field in the world presently can claim to be free of technological touch; and with technology, can achieve to any height. (Annaraja, 2015). Computer science in education, therefore is the process of teaching students how to use and understand the use of technology but not limited to computers.

Solving problems, writing algorithms, codings and decodings are the basic concepts in learning computers. It makes the students to put in keen analytic skills and decision making. Teachers can only understand strengths and weakness of the students when they are engaged with problems. In this case, students are assisted to work with computer related problems, debug and detect the faults.

Computer programs have the following qualities:

- Reliability: how often the results of a program are correct. This depends on conceptual correctness of algorithms, and minimization of programming mistakes, such as mistakes in resource management (e.g., buffer overflows and race conditions) and logic errors (such as division by zero or off-by-one errors).
- Robustness: how well a program anticipates problems due to errors (not bugs). This includes situations such as incorrect, inappropriate or corrupt data, unavailability of needed resources such as memory, operating system services and network connections, user error, and unexpected power outages.
- Usability: the ergonomics of a program: the ease with which a person can use the program for its intended purpose or in some cases even unanticipated purposes. Such issues can make or break its success even regardless of other issues. This involves a wide range of textual, graphical and sometimes hardware elements that improve the clarity, intuitiveness, cohesiveness and completeness of a program's user interface.
- Portability: the range of computer hardware and operating system platforms on which the source code of a program can be compiled/interpreted and run. This depends on differences in the programming facilities provided by the different platforms, including hardware and operating system resources, expected behavior of the hardware and operating system, and availability of platform specific compilers (and sometimes libraries) for the language of the source code.
- Maintainability: the ease with which a program can be modified by its present or future developers in order to make improvements or customizations, fix bugs and security holes, or adapt it to new environments. Good practices during initial development make the difference in this regard. This quality may not be directly apparent to the end user but it can significantly affect the fate of a program over the long term.

II. Teaching Computer Programming

The teaching of computer programming is one of the greatest challenges that have remained for years in Computer Science Education. A particular case is computer programming course for the beginners and the evidence is seen in the failure rates in programming courses. It is an established fact that computer does not do anything on its own, it is guided by written codes (programs). Computers are everywhere today just because of programs (codes) generated to solve problems in various fields of human endeavour. Meijers, Gabbay, Thagard and Woods (2009) defined computer programming as the process of designing, writing, testing debugging/troubleshooting, and maintaining the source code of computer programs. For a person to be proficient in programming, essential skills such as learning the language, composing new programs, debugging, understanding, reusing and integrating existing programs are essential (Isong, 2014). The teaching and learning of programming poses one of the greatest challenges in the area of Computer Science Education.

There is no doubt that programming is difficult despite the use of some good methods in teaching and

learning, and the authors stand to say that one of out of the numerous reasons why these problems continue to exist in our educational system is the teaching approach used in delivering Computer programming to the students.

III. Issues worth considering in Programming

Learning to write computer programs will certainly require more than just acquiring skills, it will be greatly affected by students self-efficacy that will influence the use of cognitive strategies while solving problems, the amount of effort expended, the type of coping strategies adopted, the level of persistence in the face of failure, and the ultimate performance outcomes (Wiedenbeck, LaBelle, Kain, 2004). Programming requires the use of strong cognitive skills such as reasoning, problem solving and planning. A programmer forms abstract representations of a process, expresses them in the form of logic structures, and finally translates them into correct code using the formal language outcomes (Wiedenbeck, LaBelle, Kain, 2004).

Al-Imamy and Alizadeh (2006) noted that the three pedagogical goals in teaching a programming language are language syntax, developing program design and creative thinking.

IV. Factors that Affect the Learning of Programming

a). Learning style:- This is a chosen learning approach in which students or learners finds most suitable for them. While some students are very visual, some are more auditory.

Selecting a particular learning style, or a particular form of motivation, may allow a student to acquire programming skill quickly and easily. Opposed to that, if a student adopts the wrong style or lacks the motivation, he may find learning to program difficult (Hu, 2003, Jenkins, 2002).

Jenkins (2002) classified learning styles into two, namely: deep and surface approach.

Deep learning refers to gaining understanding of a topic, while surface learning concentrates on memorising the facts. In programming, surface learning can be used for memorising the language's syntax, but deep learning is crucial, in addition to surface learning, for gaining a true understanding of programming logic and consequently a true competence in programming.

b). Motivation:- This is one other factor that affects learning efficiency and it can be Extrinsic (external), it could be Intrinsic (inborn) or social. Extrinsic motivation comes from expected external rewards, such as financial benefits, intrinsic motivation comes from within, and social from a desire to please a third party (family, teacher, friends, etc.). (Jenkins, 2001, 2002).

V. Approaches to Teaching Programming

Papp-Varga, Szlavi, and Zsako (2008) in one of their classifications, divides computer programming teaching methods according to their orientation: statement-oriented, using as a tool, software technology-oriented, task type-oriented, language-oriented, action-oriented and sample task-based. By this classification different methods cover the chosen programming language to the different extents, and teach its elements in a different order depending on the set goals. Statement oriented methods, as such, see the programming language as set of statements, and all individual elements are then taught in a certain fixed order. Task type oriented methods introduce practical problems and then present language elements in such order in which they are needed to solve those tasks. "Using as tool" methods introduce a programming language only to a necessary extent, considering a different primary goal, such as database teaching.

Teaching programming can also be divided on bottom-up and top-down approach. Bottom-up approach primarily focuses on teaching the details of syntax and individual programming language elements first. After individual elements have been taught, more complex constructs are considered. Top-down approach starts with understanding the abstractions regardless of their physical implementation. After students understand these abstractions and their purpose, implementations are being taught (Reek, 1995).

Looking at the difficulties that programming presents to students, it is therefore imperative to find and implement suitable teaching strategy so as to ensure student's success in mastering the course.

Computer programming is therefore an indispensable skill that must be mastered by anyone with interest in computer science. Normally, in teaching computer programming, students will first be introduced to the concept of programming and data structure where they are taught on how to analyze problems, use specific techniques to represent the problem solution and validate the solution. Next the learners are required to convert the problem solution into a program using a specific programming language.

Most programming courses are taught using the traditional approaches including a blend of lectures, reading and practical sessions (Gray, Boyle & Smith,

1998). The environments for these types of approaches will only produce students who are passive information receivers, allow minimal interaction between teacher and students especially when a large group of students is involved. Gage and Berliner (1992) also argued that this type of lecturing is not appropriate if specific goals and objectives need to be addressed, need long period of information retention, the learning materials are complex and abstracts, students participation in class are essential to achieve learning objectives and higher level of cognitive objectives (analysis, synthesis and evaluation) are the purpose of the instruction.

VI. Challenges of Teaching Computer Programming

Numerous challenges affect the teaching and learning of introductory Computer Programming course in most institutions of higher learning. These challenges are known to have some serious impact on both the teachers and students. These challenges include students' under-preparedness, lack of experienced CP teachers, lack of infrastructures such as well-equipped computer laboratory, lack of resources, and so on.

Computer programming is termed difficult especially for First year Computer Science students who are mostly students that have just left the secondary schools where Computer Programming is neither taught nor computer used. Reasonable number of the students have never seen, use or touch the computer since their life time. Consequently, this poses a serious challenge to the teaching and learning activities. The challenge is aggravated by the traditional objectivist lecture-based approach that is used in teaching the course. By using the traditional objectivist lecture-based approach, the teacher is the primary source of knowledge and learning is seen as an information transmission that originates from the teacher's knowledge to the students, driven by direct lecture (Wulf, 2005). Lectures take place only in classrooms and the computer laboratories are rarely used. Thus, with the unprepared nature of these students, lecturing programming in the classroom alone is challenging and sometimes it appears as if German language is used in teaching English language (Isong, 2014). This makes it a Herculean task for most students to use the computer or understand the function of a code and how they are written.

Computer Literacy

Computer literacy poses one of the problems confronting the teaching and learning of CP in most universities. In the university system, aside some of the students who started their first year after they have undergone a one year remedial programme, it is believed that majority of other students in their first year are from the secondary schools where CP was never offered. In this case, they have not seen, touch or even operate computers before. Thus, they are only touching computers for the first time in their programming class. This however, makes it difficult for the students to operate the computer, understand programming concept and structures, syntax and semantics of the language, the programming environment and actively participate in CP classes (Isong, 2014).

Lack of Access to Computers and other Resources

This is another great challenge that affects teaching and learning of Computer Programming in schools. Financial constraints pose a great situation that affects teaching and learning of Computer Programming. Most students do not have personal computers or laptops of their own, the departmental laboratory is the only place where computers can be used for their practices or projects. Meanwhile, these students can only access the laboratory most times only during the day and in their practical hours. They also faced with the problem of lack of Computer Programming textbooks.

Oroma, Wanga, and Ngumbuke, (2012) states the challenges of teaching and learning computer programming as follows: inadequately trained academic and technical staff (as compared to those in developing countries), limited access to computer facilities by students, high students intake (which increases the student – computer ratio), poor learning styles of the students, school traditions, and pedagogical methods and within the natural environment there are issues to do with poor and unstable electricity, which is a threat to hardware and data, computer viruses, high cost of internet connection, and the socio-economic environment registers concerns such as the procurement of equipment tradition, the low level of income, poor living conditions, inability to secure personal computers for learning purposes.

Learning outcomes and difficulty in learning to program could be unique to some schools. In most schools, the students' learning outcomes and difficulty in learning to program is credited to factors such as poor study methods, low self-efficacy, different kind of motivation to learn the course, lack of early exposure or previous experiences with computers, learners abilities and attitudes, nature of the course itself, limited access to computer facilities such as computer labs during a time that is for their revision and image of programmers among the

students and general public are the major hinderances to effective teaching of computer science.

Inadequate Experienced Programming Teachers

Scarcity of experienced programming teachers is another factor hindering effective delivery of computer programming. Though programming has been labeled "difficult", if experienced programming teachers use good teaching methodology in delivering the lectures, it will be easy for students to grasp.

VII. Suggested Effective Programming Teaching Methods

The following pedagogical methods have been found to be very helpful in teaching and learning of computer programming. They are:

Problem-based learning

Problem-Based Learning (PBL) is a teaching method in which complex real-world problems are used as the means to promote student learning of concepts and principles different from direct presentation of facts and concepts. PBL can promote the development of critical thinking skills, problem-solving abilities, motivation and responsibility and enhances students' communication skills. It also provides the platform for students to work in groups in groups, finding and evaluating research materials, and life-long learning.

Problem-based learning (PBL) method according to Nuutila, Torma and Malmi (2005) is directed at students' own engagement in problem solving. It is anchored around the types of problems that professionals in the field encounter on daily basis. It develops higher order thinking, disciplinary knowledge and practical skills, by facing students with a problem situation, and assigning them an active role of problem solvers.

The authors further stated that PBL is implemented by seven steps method that encourages learning by encompassing students' prior knowledge about the presented topics, practical problem situations, and the required students' subsequential elaboration on materials they have learned. Students mainly work in groups. The steps are as follows: they indentify and discuss the problem, then make a list of what they already know, and what they still need to learn. That is followed by establishing the learning goals, after which the students independently study all the required material. The students then reconvene to discuss the case, and attempt to apply what they learned in order to solve the problem. Finally, they summarize their work and elaborate on their solutions.

The benefit of this teaching method is that it has been found to encourage increased retention of knowledge over the period up to several years, and students' had better results in follow up courses, compared to their colleagues who attended traditional introductory programming courses.

Puzzle-based learning (PZBL)

Puzzle Based Learning is a new teaching and learning methodology that is focused on the development of problem-solving skills. Merrick (2010), Falkner, Sooriamurthi, and Michalewicz (2010) states that Puzzle-based learning aims at teaching students critical thinking and problem solving techniques. Puzzles should generally be easy to state and remember. A problem should present a challenge to problem solver, having no obvious solution, but still showing the promise of resolution.

Puzzle solving is conducted through the following steps in computer programming courses:

Firstly, the problem is presented to the student according to content that is being taught. A complete program solution is then divided into number of program puzzle pieces depending on the desired difficulty. Minimum size for a puzzle piece is one complete line of code. Student then attempts to reconstruct the program, by selecting the correct program pieces in the correct order, after which his success is evaluated. This procedure can be guided by a teacher or by an automated system (Yoneyama, Matsushita, Mackin, Ohshiro, Yamasaki, and Nunohiro, 2008).

An empirical study conducted by Merrick (2010), during one introductory programming course where PZBL method was implemented, showed that students interest and scope for active participation in the programming course was significantly increased.

Pair programming (PB)

Pair programming (PP) is a programming style or a software development technique in which two programmers work together on one workstation or system, on the same code. They both continuously participate in the design, development and testing of that code. The programmers that work as a pair assume two roles, one of the driver and second of the navigator.

The driver is in control, typing the actual code, while navigator observes his work, watching for potential errors, offering suggestions, and devising alternatives. They switch roles at the regular intervals, ensuring that both programmers continually provide the same amount of effort. Zacharis (2011) suggest that the software, that was developed using pair programming method, is usually completed in less time then when individual students code it, is

better designed and has less errors. Students are also better motivated to stay on task, have more confidence in their solutions, and show a positive attitude towards collaboration. Knowledge is constantly shared between pair programmers, whether in the industry or in a classroom, many sources suggest that students show higher confidence when programming in pairs, and many learn whether it be from tips on programming language rules to overall design skill. Pair programming allows programmers to examine their partner's code and provide feedback which is necessary to increase their own ability to develop monitoring mechanisms for their own learning activities.

However, studies carried out by Chaparro, Yuksel, Romero, and Bryant (2005) have shown that, despite all its benefits, pair programming can sometimes be irritating and exhausting (Chaparro, Yuksel, Romero, Bryant (2005). It is suggested that a difference in skill level between paired students strongly affects their collaboration, and that an appropriate care when organizing the pairs of students is necessary. Virtual pair programming is a form of pair programming that eliminates the need of two programmers sitting at the same physical location. Their collaboration is realized by using online tools that integrate desktop sharing and real time communication. The study carried out by Zacharis (2005) showed that it is an effective pedagogical tool for flexible collaboration.

Prerecorded Lectures (PL)

Prerecorded lectures (PL) in form of carefully prepared and technically focused multimedia recordings that are kept online, comprise of narrated slides from the lecture notes, and are meant to supplement the conventional lectures. PL aim to confront the issue of limited semester timetable, helping those students who fail to absorb the presented course material at sufficient rate. (Smith and Fidge, 2008).

They provide students with a possibility to review all the concepts they may have missed or not understood well enough while attending the original lectures. Unlike classic textbooks, PL better separate the important concepts from less important ones, and can be more easily refined over time to better stress those particular concepts that students had issues with over past semesters (Smith and Fidge, 2008). Prerecorded lectures provides students with modules allows them to review a recorded lecture multiple times and learn the course material at their own pace. Use of the modules also frees up classroom time that can then be used for active learning exercises and detailed application of the material. Students believed that combining pre-recorded modules with class exercises allowed them to become more actively engaged in the learning process and develop better understanding of the course material.

Smith and Fidge (2008) conducted a research among students that used 5 to 15 minutes long PL, each focusing on particular topic of the course, it was found that PL made no significant difference on students final grades. It was also found that part of the students, who have seen the PL only as an optional material, have never accessed any of the PL online. Overall, however, students' feedback on the PL was very positive, and most of them stated that PL helped them better understand some programming concepts.

Game-themed programming (GTP)

Game-themed programming (GTP) is a teaching approach which integrates simple interactive graphic programs into introductory programming courses. The aim of this approach is to make students learn about abstract programming concepts by exploring and programming small game applications (Sung, Hillyard, Angotti, Panitz, Goldstein, Nordlinger, 2010)

The major objective of GTP is not teaching students to build computer games, but teaching them programming concepts through understanding how the games work. Any assignment that would, in a classical approach, be a simple console assignment, is reworked so it becomes a simple game assignment, and that the devised game logic is based on targeted programming concept. Game-themed assignments, which are given to students, include the descriptions of the tasks that need to be completed and starter projects that consist of provided graphics and functional user interaction modules. Students are then required to complete the assigned projects by filling in the missing relevant concepts.

In a study conducted by Sung, Hillyard, Angotti, Panitz, Goldstein, Nordlinger, (2010), it revealed that success rates in GTP classes were higher than in normal classes. Students reported that they had to spend more time to understand the game assignments than they would require to understand classical console assignments, but when they finally understood them, their motivation and enthusiasm increased.

Teacher's task is to select an appropriate teaching method, or a blend of methods, to devise assignments for students accordingly, and after presenting the concepts in the way he/she has chosen, to shift the focus on students, motivating them to engage in assignments he has devised.

Dalton and Goodrum (1991) have suggested that computer programming and problem solving strategy instruction, when used together may provide an effective means of teaching transferable problem solving skills.
Maheshwari (1997a) also suggested that programming lessons should employ systematically designed direct instruction activities, rich in feedback and practice opportunities. Programming activities should be designed to encourage the application of problem solving strategies such as planning, simplification and modeling. She also stated that lessons should quickly develop a rudimentary mastery of language syntax and move quickly to produce application and problem solving. In other words, teaching programming should be interesting, motivating and stimulating for both students and lecturers.
The students need to understand how to interpret the given problem before they can represent the correct solution and effectively use specific tools or techniques. The later skills can be acquired by doing a lot of practices in problem solving that involved planning, logical thinking and reasoning strategies. However, mastery in the reasoning and problem solving skills does not necessary mean that students are able to write good computer program as writing programming languages

VIII. Conclusion

Computer programming has become so important that it cannot be neglected if we must move ahead. Hence, serious efforts must be made to improve on its teaching and learning. Computers are not humans and cannot do anything on its own, it must be instructed to do anything through programming.
Teaching and learning of computer programming has been a difficult task especially in the developing countries due to several reasons or factors. Some are student centered while some are teacher centered. Using the correct teaching method in teaching programming remains the best bet for effective transfer of programming knowledge and skills. On the part of the students, prior knowledge of computer usage, critical thinking skills, appropriate learning styles, possession of a personal computer and good computer programming textbooks will be help the students in no small measure.

IX. Recommendations

The following recommendations are hereby made for effective computer programming delivery
1). Appropriate teaching methodology must be used if students are to encouraged to undertake computer programming courses.
2). Students should be exposed to the use of computers right from their secondary schools so that it will not be new to them in their advanced level.
3). Experienced computer programmers should be recruited to teach in schools and not those with “I can teach mentality”.
4). The price of personal computers should be subsidized so that students can afford to have theirs.
5). The teaching of computer studies in secondary schools should be made compulsory so that it will not remain a mystery to them in first year of tertiary education.

X. REFERENCES

Annaraja, P. (2015). Teaching of Computer Science B.Ed. I Year. The Syllabus adopted from 2015-16 onwards. SCAD college of Education Cheranmahadevi, Ttirunelveli District.

Al-Imamy S, Alizadeh J. (2006) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Bebbington, S. (2014). What is coding? Retrieved from https://en.wikipedia.org/wiki/Computer_programming

Carter, J and Boyle, R. (2002). Teaching Delivery Issues -Lessons from Computer Science. *Journal of Information Technology Education. Volume 1 No. 2*

Chaparro E A, Yuksel A, Romero P, Bryant S (2005). Factors Affecting the Perceived Effectiveness of Pair Programming in Higher Education, *Proceedings 17th Workshop of the Psychology of Programming Interest Group.*

Gao, J. and Hargis, J (2010). Promoting Technology-assisted Active Learning in Computer Science Education. *The Journal of Effective Teaching. An online journal devoted to teaching excellence. Vol. 10, No. 2.*

Hu, M. (2003) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from

archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Ismail, M.N, Ngah, N.A, Umar, I.N. (2010). Instructional Strategy in the Teaching of Computer Programming: A Need Assessment Analyses. TOJET: *The Turkish Online Journal of Educational Technology. volume 9 Issue 2*

Isong,B. (2014). A Methodology for Teaching Computer Programming: first year students' perspective. *International Journal of Modern Education and Computer Science,* 9, 15-21

Jenkins T. (2002) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Meijers A W M, Gabbay D M, Thagard P, Woods J (2009) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Oroma, J.O., Ngumbuke, F. and Wanga, H.P. (2012). Programming in Developing Countries: Lessons from Tumaini University. Retrieved from https://www.researchgate.net/publication/267926470_Challenges_of_Teaching_and_Learning_Computer_Programming_in_Developing_Countries_Lessons_from_Tumaini_University

Papp-Varga Z, Szlavi P, Zsako L (2008) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Reek, M. M. (1995) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Smith G, Fidge C. (2008). On the efficacy of prerecorded lectures for teaching introductory programming, *Proceedings of the tenth conference on Australasian computing education, Vol. 78.*

Sung K, Hillyard C, Angotti R L, Panitz M W, Goldstein D S, Nordlinger J (2010). Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses, IEEE Transactions on Education, Vol PP, Issue 99.

Szlávi, P. and Zsakó, L. (2003). Methods of teaching programming. Retrieved from https://www.researchgate.net/publication/235925815.

Wiedenbeck S, LaBelle D, Kain V N R (2004) in Mohorovicic, S. and Strcic, V. (2011). An Overview of Computer Programming Teaching Methods. Retrieved from archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238

Zacharis N. (2009). Evaluating the Effects of Virtual Pair Programming on Students' Achievement and Satisfaction, *International Journal of Emerging Technologies in Learning (iJET), Vol. 4, No. 3.*