Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 8, August 2014, pg.547 – 554

RESEARCH ARTICLE

Using the Literature to Develop a Preliminary Conceptual Model for the Student Success Factors in a Programming Course: Java as a Case Study

[1]Salam Abdulabbas Ghanim, [2]Nassir Jabir Al-khafaji

[1]Directorate of Education/ Ministry of Education, Dhi-Qar, Iraq

[2]School of Computing, Universiti Utara Malaysia, Kedah, Malaysia

Salam_a201080@yahoo.com, nassirfarhan@yahoo.com

Abstract— The complexity and difficulty ascribed to computer programming have been asserted to be the causes of its high rate of failure record and attrition. It is opined that programming either to novice, middle learner, and the self-branded geeks is always a course to be apprehensive of different studies with varying findings. Studies on factors leading to the success of programming course in higher institutions have been carried out. Many Universities have greatly high failure rates, in practically, Java. This really motivates this study, which aims at discovering the factors affecting the success of programming courses.

Index Terms— Conceptual Model, Success Factors, University, Student, Object Oriented Programming

I. INTRODUCTION

Modern curriculum needs to emphasize on the development of programming skills for citizens of a technological society [1]. Programming is a cognitive activity that requires abstract representations and logical expressions. The program must translate abstract representations into correct codes by using a formal language to create, modify, reuse, or debug a program [2]. Furthermore, programming is often viewed as a problem-solving activity rather than a linguistic activity, often ignoring the fact that programming languages are a case of formal languages. The interpretation of formal languages is unique for every individual.

Programming skills are an essential part of computer science (CS) and information technology (IT) courses [3]. Robins, Rountree, and Rountree [4] argued that programming skills are useful in programming knowledge and strategies, such as program generation and comprehension. Programming can also lead to a rewarding career, such as an analyzer, programmer, or debugger.

© 2014, IJCSMC All Rights Reserved

Zdancewic and Weirich [5] stated that programming is a conceptual foundation in the study of computations. Programming is a prerequisite for almost every other course in CS. Renumol, Jayaprakash, and Janakiram [6] said that "programming is the process of writing, testing and debugging of computer programs using different programming languages." However, according to Schreiner [7], a program is the formal description of a method that solves a particular problem.

Programming languages has two basic levels: a high-level languages, which are classified into three groups, namely, procedural (C, C++, Visual Basic, and Java), non-procedural (LISP and PROLOG), and, problem oriented (MATLAB, MATHEMATIC, and LATEX); a low-level languages, such as machine language and assembly language [8]). Matravers [9] argued that the low-level representation of a central processing unit instruction set is known as the machine language of a computer. Thus, directly writing instructions in binary form is difficult.

This study focuses on Java as OOP. According to Bennett, Fisher, and Lees [10], no differences exist between OOP and the restructuring of a high-level world view where the object in OOP has attributes that are the same with attributes of the object in the real world (i.e car has name, color, model, etc). Furthermore, Poo and Ashok [11] stated that OOP set data and operations into units called objects and allowed objects to be combined into systemic networks to build a program. Objects and their interactions are the main elements of program design in OOP. Each object has a state (data) and a behavior (operations on data).Thus, Objects in OOP are not much different from ordinary physical objects.

Educating novices on programming has been considered as a big challenge since the early 1970s [12] [13] [14] [4].Teaching programming is considered one of the seven grand challenges in computing education [15] . Sharp and Schultz [16] found that learning OOP is difficult for students because it requires skills of comprehension and memorization abilities; the latter involves high-level abilities, which requires additional skills such as abstraction, encapsulation, polymorphism, and inheritance

II. LITERATURE REVIEW

A. Object Oriented Programming

Programming knowledge includes skills and concepts such as problem investigation, problem-solving design, transformation of the design into code and data structure by writing a highly constrained language, and verification of the validity of the program [17]. In recent years, OOP has become the most influential programming paradigm. OOP is widely used in education and different industries; furthermore, almost every university includes object orientation in the curriculum [18].

Learning to program is notoriously difficult. For instance, Bergin and Reilly [40: 293] noted that "it is well known in the computer science education (CSE) community that students have difficulty with programming courses and this can result in high dropout and failure rates." At the same time, according to many researchers, teaching programming to novices has been considered a big challenge for almost 40 years (e.g. [14] [4]). Teaching programming is considered one of the seven grand challenges in computing education [15]. According to Diederich [19], the relevant elements in teaching can often be described by the didactic triangle, see Figure 1.

Unfortunately, more studies have concentrated on the teacher as the substantial factor, and few studies have focused on the students and content. Many published research materials on the Java programming language mostly focus on technology issues and related enhancements. Therefore, this study seeks to fill this gap in literature by identifying factors that affect the teaching of programming from the perspective of students.

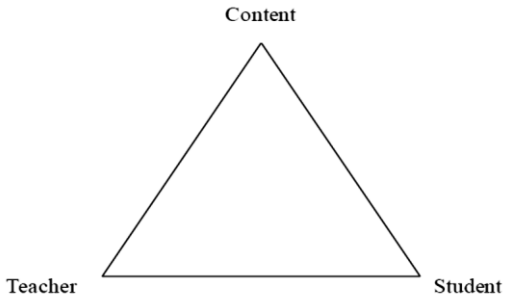

Figure 1. The didactic triangle [19].

© 2014, IJCSMC All Rights Reserved

Madsen and Møller-Pedersen [20: 16] defined OOP as follows: "a program execution is regarded as a physical model, simulating the behavior of either a real or imaginary part of the world." OOP is a clever concept and has become a very common term (Henderson & Zorn, 1994[21]). As stated beforehand, OOP was used as the first language in most universities, in particular Java programming (Schulte & Bennedsen, 2006 [22]). However, numerous studies have focused on how to develop OOP programming learning.

a. Java Programming

Sivasakthi and Rajendran [18: 1] stated the following: "Java programming is popular both in Academia and IT Industry. Further, it is the maximum usage of programming across the world." Moreover, given the new possibilities provided by Java for the web, the Java paradigm has received considerable attention. Thus, many universities and colleges have introduced Java into their undergraduate and postgraduate CS curriculum [23]. Thus, the teaching and learning Java programming in academia have become a great responsibility. Madden and Chambers [24] added that the Java programming language is very well established and is often the first object-oriented (OO) language taught to students.

Despite the popularity of programming languages such as Java, issues still exist on the suitability of these languages for education, particularly in the introduction of programming to novices (for instance [23][25] [26]). Pears et al. [27] stated that Java is not designed for educational purposes compared with Python, Logo, Eiffel, and Pascal.

This study aims to identify the factors that cause the learning difficulties of students with regards to Java as OOP. Java has become the most influential programming paradigm in recent years. Although empirical studies of programmers and programmer comprehension have been conducted with regards to procedural and OO languages, few studies have been conducted to discover the individual traits cause the most difficulty to novice programming students [28]

B. Previous Study

The high dropout and failure rates in programming subjects have drawn the attention of researchers. A number of papers have also been written to address problems that occur when teaching Java. However, most of these studies have focused more on the teachers than on the students. Therefore, this section elaborates on few studies that are related to this field. Such as the one that was conducted by Byrne, Catrambone, and Stasko [29] who used two experiments designed to test whether animating algorithms will assist students to learn algorithms effectively. However, this study focuses only on software visualization (teaching tools). While, Wilson and Shrock [30] stated that many factors affect the success or failure of students in programming.

Further, the study conducted by Byrne and Lyons [31] focused on the BASIC programming language and their data has been gathered from academic records. In contrast, this research concentrates on the OOP as noted by Wiedenbeck [32] the choice of programming language affects the understanding of programming.

On the Other hand, Milne and Rowe [28] have investigated C++ and asked both tutors and students on the individual concepts of the programming language they strive to teach and learn, the conclusion of their study is the motivation to design a program visualization tool. As well as, they focused on participants who have experience in programming languages, while, this study deals with the novice students.

Wiedenbeck and Labelle [33] investigated the combined effects of mental model, self-efficacy, and prior experience on programming learning. However, the respondents came from different disciplines. Most respondents are not involved in CS.

In addition, Bennedsen and Caspersen [34] focused on learning OOP. Their study depended on the perspectives of lecturers and that of the university administration. By contrast, this research concentrated particularly on the perspective of students. On the other hand, study by Caspersen and Kölling [35] aimed to assist novice programmers learn better and faster. In the same time, laying the foundation for a thorough treatment of the aspects of software engineering. Their study did not identify the factors that affect how students learn programming and instead focused on the programming process (Concepts of the program).

© 2014, IJCSMC All Rights Reserved

III. PRELIMINARY CONCEPTUAL MODEL

Most novice programmers still struggle to become proficient in the subject. Therefore, this study focuses on the following factors (Fear, Object Oriented Programming Concepts and Motivation) that affected the academic success of novice students in computer programming based on literature review. Eventually, based on these factors this study sought to develop Preliminary Conceptual Model as illustrated in Figure 2:

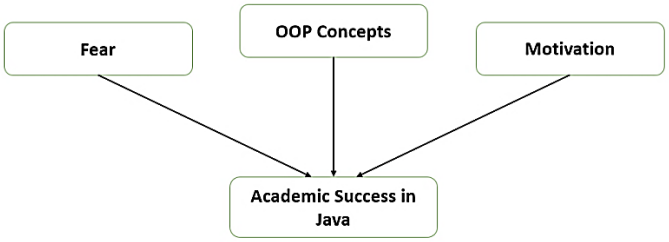

Figure 2. Preliminary Conceptual Model for Student Success Factors

A. Fear

The teaching and learning processes in the CS and IS fields have received significant research attention because of the high attrition rates of these courses [4]. However, the educational environment of these courses remains a global problem in the field of computer programming [36].

Study by Rogerson and Scott [37] defined fear as the lack of appreciation or interest of students toward the programming subject. This term may also refer to the apprehension or the lack of confidence of these students on their programming knowledge. Programming is defined in this study as the full cycle of systems development, including the coding process, the use of basic theoretical concepts, and preparation of the final product for implementation [38].

Rogerson and Scott [37: 148] defined this fear as "experiencing a lack of confidence or apprehension regarding their ability to code or program". Several researchers have used the same term to describe the anxiety that some students feel when developing a program and to describe their feelings of discomfort that may reduce their interest in the subject [39]. Bergin and Reilly [40] stated that such feeling of discomfort will discourage programming related inquiries and discussions from these students.

B. Object Oriented Programming Concepts

Students are greatly challenged by several elements in the Java programming language. The inclusion of Java in programming courses has been the subject of several studies and experience reports over the previous decade [18] [41]. These reports suggest that the use of diagrammatic representations can help students improve their understanding of OOP, such as UML or other analogous notations [42].

Sicilia [43] argued that programming instructors should carefully help their students in comprehending the OO concepts and in translating the conceptual models into Java programs. The OOP learning of these students is also hindered by several factors, such as their associations, generic containers, and differences between interfaces and classes. Moreover, Madden and Chambers [24] asserted that, comprehension of a list of broad Java language tools such as (e.g. syntax, file handling, inheritance, Appletviewer, JCreator, GUI programming, etc) help to understanding OOP concepts. Many researchers have explained the OOP concepts as shown below:

a. Object

Object refers to the main component of the OO paradigm that is used for carrying out specific tasks [44]. Actual examples of an object include a bus, a book, or a student. Therefore, people think about, identify, act upon, or assign concepts to several objects on a daily basis [45].

© 2014, IJCSMC All Rights Reserved

b. Class

Students must clearly differentiate the concept of "object" from the concept of "class" [46]. The latter refers to a general category, whereas the former refers to a specific instance [45]. Objects are grouped together into classes that specify the type of an object, whereas a class can be used as a template for a potential object [47].

c. Attributes and Methods

Both attributes and operations are equally important in the OO approach. The former refers to the descriptive properties of an object that represent its state, whereas the latter refers to an operation that determines the behavior of an object or what the object can do [48].

d. Constructors and Destructors

Constructors and destructors are special methods that play important roles in OOP. Constructors are used when creating new objects for the allocation of memory and the initialization of variables. Sebesta [49] referred that, Delphi uses the "create" constructor to create an object. Additionally, he stated that, all objects in Java are explicit heap dynamic (i.e., created explicitly on the heap during runtime) and are allocated into the new operator. Destructors are used to reclaim the heap storage and to destroy objects. Instead of using a destructor, Java uses an implicit garbage collection process that does not require the programmer to create a code for the destructor [48].

e. Abstraction and Associations

Abstraction refers to the ability of an individual to define and use variables and operations that ignore several details. Abstraction aims to simplify the presentation of entities and to reduce their complexity during the programming process [49]. Abstraction is classified into process abstraction and data abstraction. The former refers to the calling of a subprogram (method/procedure/function) without providing its details, whereas the latter refers to the declaration of the type and the operations in objects that are contained in a single unit, which restricts data access by sending messages to the methods [50].

f. Polymorphism and Dynamic Binding

Polymorphism refers to the provision of multiple forms and methods. When used in the OOP context, this term implies that different objects may respond individually to the same message. Therefore, polymorphism may be used to indicate different implementations [51]. Polymorphism also supports greater abstraction wherein a single message can evoke different behavior [52].

C. Motivation

Helme and Clarke [53] stated that students need motivation (the will to learn) and skills (capability) in order to be successful in their respective fields. Williams [54] argued that the learning methods, motivation, and expectation of students can significantly affect their learning. Several studies have identified the motivation and attitude of these students to learning as the core influential factors to their successful learning [55] [56] [4] [57]. Moreover, Jenkins [58]; Bergin and Reilly [40] pointed that, the motivation can encourage the students to learning programming language well. In this case, motivation can be divided into intrinsic, extrinsic, and achievement motivation [59].

- Intrinsic motivation is present when the individual is interested and curious about the activity that he or she is currently performing;
- Extrinsic motivation is present when the individual anticipates a reward after successfully completing the activity; and
- Achievement motivation is observed when the performance of an individual is better than that of his or her peers.

© 2014, IJCSMC All Rights Reserved

Carbone, Hurst, Mitchell, and Gunstone [60] found that intrinsically motivated students generally display higher programming capabilities, whereas externally motivated (i.e., passing the course) or achievement-motivated (i.e., obtaining higher marks) students do not cognitively engage themselves into the subject.

IV. CONCLUSION

In general, learning to program is difficult for many students. Although several factors that affect learning to program have been identified over the years, we are still far from a full understanding of why some students learn to program easily and quickly while others flounder. Besides, previous researches have not mentioned all the important factors. Based on the current research, three factors that may affect learning to program generally, and Java especially are Fear, Object Oriented Programming Concepts and Motivation. The limitations of this study are concentrated only on the literature review. Consequently, the future work must investigate these factors in real life.

REFERENCES

[1] B. Pejcinovic, M. Holtzman, M. Chrzanowska-Jeske, and P. K. Wong, "Just because we teach it does not mean they use it: Case of programming skills," in *Frontiers in Education Conference, 2013 IEEE*, 2013, pp. 1287-1289.

[2] S. Wiedenbeck, "Factors affecting the success of non-majors in learning to program," in *Proceedings of the first international workshop on Computing education research*, 2005, pp. 13-24.

[3] R. Mason, G. Cooper, and M. de Raadt, "Trends in introductory programming courses in Australian universities: languages, environments and pedagogy," in *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123*, 2012, pp. 33-42.

[4] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, pp. 137-172, 2003.

[5] S. Zdancewic and S. Weirich, "Programming Languages and Techniques," 2013.

[6] V. Renumol, S. Jayaprakash, and D. Janakiram, "Classification of cognitive difficulties of students to learn computer programming," *Indian Institute of Technology, India*, 2009.

[7] W. Schreiner, "Introduction to Programming," 2011.

[8] V. Rajaraman, "Programming languages: A brief review," *Resonance-Journal of Science Education*, vol. 3, pp. 43-54, 1998.

[9] J. Matravers, "Introduction to computer systems architecture and programming," 2011.

[10] G. Bennett, M. Fisher, and B. Lees, "Object Oriented Programming with Objective-C," in *Objective-C for Absolute Beginners*, ed: Springer, 2011, pp. 87-102.

[11] D. Poo, D. Kiong, and S. Ashok, *Object-oriented programming and Java*: Springer, 2007.

[12] E. W. Dijkstra, *Notes on structured programming*: Technological University Eindhoven Netherlands, 1970.

[13] D. Gries, "What should we teach in an introductory programming course?," in *ACM SIGCSE Bulletin*, 1974, pp. 81-89.

[14] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," *ACM SIGCSE Bulletin*, vol. 33, pp. 125-180, 2001.

[15] A. Mcgettrick, R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove, and K. Mander, "Grand challenges in computing: Education—a summary," *The Computer Journal*, vol. 48, pp. 42-48, 2005.

[16] J. H. Sharp and L. A. Schultz, "An exploratory study of the use of video as an instructional tool in an introductory C# programming course," *Information Systems Education Journal*, vol. 11, p. 33, 2013.

[17] N. S. Herman, S. B. Salam, and E. Noersasongko, "A Study of Tracing and Writing Performance of Novice Students in Introductory Programming," in *Software Engineering and Computer Systems*, ed: Springer, 2011, pp. 557-570.

[18] M. Sivasakthi and R. Rajendran, "Learning Difficulties of'Object-oriented Programming Paradigm Using Java': Students' Perspective," *Indian Journal of Science and Technology*, vol. 4, pp. 983-985, 2011.

[19] J. Diederich, "Didaktisches Denken," *Eine Einführung in Anspruch und Aufgabe*, 1988.

[20] O. L. Madsen and B. Møller-Pedersen, "What object-oriented programming may be-and what it does not have to be," in *ECOOP'88 European Conference on Object-Oriented Programming*, 1988, pp. 1-20.

[21] R. Henderson and B. Zorn, "A comparison of object-oriented programming in four modern languages," *Software: Practice and Experience*, vol. 24, pp. 1077-1095, 1994.

[22] J. Bennedsen and M. E. Caspersen, "Abstraction ability as an indicator of success for learning object-oriented programming?," *ACM SIGCSE Bulletin*, vol. 38, pp. 39-43, 2006.

© 2014, IJCSMC All Rights Reserved

[23] S. Hadjerrouit, "Java as first programming language: a critical evaluation," *ACM SIGCSE Bulletin*, vol. 30, pp. 43-47, 1998.
[24] M. Madden and D. Chambers, "Evaluation of student attitudes to learning the Java language," in *Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002*, 2002, pp. 125-130.
[25] R. Close, D. Kopec, and J. Aman, "CS1: perspectives on programming languages and the breadth-first approach," in *Journal of Computing Sciences in Colleges*, 2000, pp. 228-234.
[26] D. Clark, C. MacNish, and G. F. Royle, "Java as a teaching language—opportunities, pitfalls and solutions," in *Proceedings of the 3rd Australasian conference on Computer science education*, 1998, pp. 173-179.
[27] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," in *ACM SIGCSE Bulletin*, 2007, pp. 204-223.
[28] I. Milne and G. Rowe, "Difficulties in learning and teaching programming—views of students and tutors," *Education and Information technologies*, vol. 7, pp. 55-66, 2002.
[29] M. D. Byrne, R. Catrambone, and J. T. Stasko, "Evaluating animations as student aids in learning computer algorithms," *Computers & education*, vol. 33, pp. 253-278, 1999.
[30] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: a study of twelve factors," in *ACM SIGCSE Bulletin*, 2001, pp. 184-188.
[31] P. Byrne and G. Lyons, "The effect of student attributes on success in programming," in *ACM SIGCSE Bulletin*, 2001, pp. 49-52.
[32] S. Wiedenbeck, V. Ramalingam, S. Sarasamma, and C. Corritore, "A comparison of the comprehension of object-oriented and procedural programs by novice programmers," *Interacting with Computers*, vol. 11, pp. 255-282, 1999.
[33] S. Wiedenbeck, D. Labelle, and V. N. Kain, "Factors affecting course outcomes in introductory programming," in *16th Annual Workshop of the Psychology of Programming Interest Group*, 2004, pp. 97-109.
[34] C. Schulte and J. Bennedsen, "What do teachers teach in introductory programming?," in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 17-28.
[35] M. E. Caspersen and M. Kölling, "A novice's process of object-oriented programming," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 892-900.
[36] J. Mead, S. Gray, J. Hamer, R. James, J. Sorva, C. S. Clair, and L. Thomas, "A cognitive approach to identifying measurable milestones for programming skill acquisition," in *ACM SIGCSE Bulletin*, 2006, pp. 182-194.
[37] C. Rogerson and E. Scott, "The fear factor: How it affects students learning to program in a tertiary environment," *Journal of Information Technology Education: Research*, vol. 9, pp. 147-171, 2010.
[38] C. Bruce, L. Buckingham, J. Hynd, C. McMahon, M. Roggenkamp, and I. Stoodley, "Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university," *Journal of Information Technology Education: Research*, vol. 3, pp. 145-160, 2004.
[39] S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. De Raadt, P. Haden, J. Hamer, M. Hamilton, and R. Lister, "Predictors of success in a first programming course," in *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 2006, pp. 189-196.
[40] S. Bergin and R. Reilly, "The influence of motivation and comfort-level on learning to program," in *Proceedings of the PPIG*, 2005, pp. 293-304.
[41] S. Xinogalos, M. Sartatzemi, and V. Dagdilelis, "Studying Students' Difficulties in an OOP Course based on BlueJ," in *IASTED International Conference on Computers and Advanced technology in Education*, 2006, pp. 82-87.
[42] C. Alphonce and P. Ventura, "Object orientation in CS1-CS2 by design," in *ACM SIGCSE Bulletin*, 2002, pp. 70-74.
[43] M.-Á. Sicilia, "Strategies for teaching object-oriented concepts with Java," *Computer Science Education*, vol. 16, pp. 1-18, 2006.
[44] J. M. Garrido, *Object-Oriented Programming: From Problem Solving to Java*: Firewall Media, 2004.
[45] J. W. Satzinger and T. U. Ørvik, *The Object-oriented Approach: Concepts, System Development and Modeling with UML*: Course Technology, 2001
[46] A. Eckerdal and M. Thuné, "Novice Java programmers' conceptions of object and class, and variation theory," in *ACM SIGCSE Bulletin*, 2005, pp. 89-93.
[47] M. Weisfeld, *The object-oriented thought process*: Pearson Education, 2008.
[48] H. M. Havenga, "An investigation of students' knowledge, skills and strategies during problem solving in objectoriented programming," 2009.
[49] R. W. Sebesta, *Concepts of programming languages* vol. 4: Addison Wesley, 2002.
[50] S. R. Schach, *Object-oriented and classical software engineering* vol. 6: McGraw-Hill New York, 2002.
[51] M. A. Weiss and S. Hartman, *Data structures and problem solving using Java* vol. 204: Addison-Wesley Reading, 1998.
[52] M. B. Rosson and S. R. Alpert, "The cognitive consequences of object-oriented design," *Human-Computer Interaction*, vol. 5, pp. 345-379, 1990.

© 2014, IJCSMC All Rights Reserved

[53] S. Helme and D. Clarke, "Identifying cognitive engagement in the mathematics classroom," *Mathematics Education Research Journal*, vol. 13, pp. 133-153, 2001.
[54] K. C. Williams and C. C. Williams, "Five key ingredients for improving student motivation," *Research in Higher Education Journal*, vol. 12, pp. 1-23, 2011.
[55] A. Gomes and A. J. Mendes, "Learning to program-difficulties and solutions," in *International Conference on Engineering Education–ICEE*, 2007.
[56] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002, pp. 53-58.
[57] S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. De Raadt, P. Haden, J. Hamer, M. Hamilton, and R. Lister, "Predictors of success in a first programming course," in *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 2006, pp. 189-196.
[58] T. Jenkins, "Teaching programming–A journey from teacher to motivator," in *2nd Annual LTSN-ICS Conference*, 2001.
[59] N. Entwistle, "Motivation and approaches to learning: Motivating and conceptions of teaching," *Motivating students*, pp. 15-23, 1998.
[60] A. Carbone, J. Hurst, I. Mitchell, and D. Gunstone, "An exploration of internal factors influencing student learning of programming," in *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95*, 2009, pp. 25-34.

© 2014, IJCSMC All Rights Reserved