A Conceptual Approach for Understanding Computer Programming Skills Development

Asma Md Ali[a], Afidalina Tumian[b], Muhamad Sadry Abu Seman[a]
[a]Department of Information Systems, [b]Department of Computer Science,
Kulliyyah of Information and Communication Technology
International Islamic University Malaysia
Kuala Lumpur, Malaysia
sis_asma@iium.edu.my

***Abstract*—This paper highlights the relevant curricular and pedagogical ideas on how to evaluate students in learning computer programming. It explores the literature in the area of programming skills development towards constructive alignment in outcome-based education. Research on teaching computer programming has led to numerous practices, instructional theories, and empirical results for computer science pedagogy. The change towards constructivism seems promising in establishing a systematic computer programming skills development in the future. Three main areas that affect students learning computer programming are teaching approach, computer programming language selection, and programming development environment. The outcome of this paper is a conceptual approach that covers (1) the teaching and learning alignment as unit of analysis (2) a holistic approach on how programming skills are developed, (3) an emphasis on activities and (4) the importance of providing guidelines for lecturers.**

Keywords—Programming skills development; teaching programming approaches; programming languages; constructive alignment

I. INTRODUCTION

Computer programming is an essential skill for science and technology based degree holders. In fact, most science-based courses nowadays require students to solve work-based problems using computer programming. In a typical classroom, lectures and tutorial sessions are commonly used for teaching and learning computer programming [1]. Assignments and theory-based examinations are usually given to the students as the course assessment. However, these assessments do not encourage interactive discussion and engagement between students and lecturers. As a result, the majority of students have failed to meet the expected outcome, which is the ability to solve work-based problems through computer programming [2].

The research question formulated for this paper is how to understand programming skills development in computer programming courses at an international Malaysia public university.

II. THE NEED FOR A PROGRAMMING SKILLS DEVELOPMENT CONCEPTUAL FRAMEWORK

Choosing appropriate teaching method can influence students' performance [3]. A good selection of teaching method will increase the student understanding. Students sometimes face the problem to understand the abstract concept of programming languages [4]. Language barrier also contributes to the difficulties in learning computer programming for non-English speaking background students [4]. Difficulties to understand explanation during lecture affects students' understanding on computer programming.

The lecturer's role to facilitate learning in order to get the students involve in the class discussion is important [5]. [4] agreed on the lack of teaching skills, a poorly designed course and lack of support from the faculty are among the reasons that contribute to the failure of programming skills development. Students will lose their motivation in learning computer programming if they continue to fail in computer programming courses [6].

The balance of time between learning the syntax and hands-on activities is a major concern in programming courses that lead to the lack of programming skills [7]. There is a correlation between failures in introductory courses with programme withdrawal. Thus, the introductory programming class is very crucial for the students. Students also need to study multiple computer programming languages throughout their studies. The idea of ineffective pedagogy relates to syntactical problems and problem solving deficiencies [8].

III. CONSTRUCTIVE ALIGNMENT AND DESIGN SCIENCE RESEARCH IN INFORMATION SYSTEM RELEVANCE TO THE PROGRAMMING SKILLS DEVELOPMENT DOMAIN

A. CONSTRUCTIVE ALIGNMENT

Constructive alignment principles consist of three main elements, which are learning objectives, learning activities and learning assessment. These three principles are used to evaluate the alignment of a lecturer's designed learning outcomes, prescribed activities and assessments in order to achieve the learning outcomes. The learners create meaning from learning

978-1-5090-6255-3/17/$31.00 ©2017 IEEE

activities and assessments based on their background knowledge and experiences as suggested by the constructivist approach in learning [9].

The literatures relevant to programming skills development have been evaluated through the challenges and potential solution in improving the process of learning and teaching programming courses in higher education. Three common themes were identified which are teaching approaches, programming languages chosen and programming environment.

B. TEACHING APPROACHES

There are three categories of teaching approaches used in programming skills development.

The first approach in teaching is the ability to balance out the time for theory and practical knowledge delivery. For example, students claimed that they were under pressure in introductory course with four hours of lectures and exercises per week with no mandatory homework. The students could not comprehend the course contents properly before they move on to another topic. Thus, student-learning time has been modified to include mandatory homework that encourages students to extend their learning at home [10].

The second approach is the adaptation ability of lecturer's to explain computer-programming concepts in various levels of understanding. For example, experienced students understood the explanation, but for average students the terminology was formal and confusing [10]. The lack of effective lecturer-to-student and student-to-student communication in sharing crucial knowledge at the right time contributes to misunderstanding computer programming concepts. Thus, the learned concepts are applied incorrectly [11]. Modification of teaching method requires several areas and was influenced by the order of the topics in computer-programming course. In addition, the lecturer needs to explore a suitable way in explaining computer programming including practical examples.

Continuous feedback and scaffolding (support) are two methods in adapting lecturers' explanation. [12] presented an extreme apprentice model that emphasizes scaffolding in combination with values and practices which provides a structure for teaching skills for building routine and best practices from the masters. Continuous feedback and scaffolding to provides sufficient support to help inefficient novices to learn computer programming. Continuous feedback allows the students to receive multi-level feedback from their lecturer and enable the lecturer to monitor, and the lecturer receives feedback by monitoring the students' progress. Scaffolding is a process in which a lecturer adds supports for students to enhance their learning and achieve to complete the tasks by conducting relevant exercises.

The third approach is modifying the teaching approach structure for students' motivation. Previous computer programming introductory courses resembled math, algebra, or similar courses that focus on formula and numbers. The structure of practical examples was using the following sequence: getting the data, data processing, and outputting the result, in numeric form. This was not very motivating for students to learn. Thus, there is a move to change to teach computer-programming through graphics using a graphical robot [10]. The visual result for introductory computer programming course contributes to a higher level of motivation for students to learn [13].

C. PROGRAMMING LANGUAGES CHOSEN

Programming language is used as a teaching tool in developing logical thinking in programming skills development. There are three categories of programming language: scripting, procedural and object-oriented.

The first category is scripting programming language. Students may encounter problems using Java and C++, which can be avoided with the use of scripting languages, JavaScript in particular [14]. JavaScript is a modern, clean programming language sufficient to cover most of what is required in an elementary course. In addition, it is a language with immediate application development as it displays the output in a standard browser using a simple editing environment. The complexity of the language is reduced as it is not as hard-rule based object oriented as Java programming. Complexity of object oriented is used when needed to solve problems rather than the beginning that the novice has to encounter [14]. The additional advantage is that it introduces the class-based approach in a light-code manner. Advance programming courses such as Web Programming courses are also available in JavaScript using Node.JS.

The second category is procedural programming language. The approach in teaching procedural programming languages such as C need to be changed from basic syntax introduction to procedural functions [15]. By using this approach, students are not introduced to the specific input output commands, different expressions, conditional expressions, loops, and etc. The students are directly introduced to the procedural approach using a simple program that is easy to follow using a function. For example, scanning a number of inputs, process the number using absolute function and print the output. Examples given to the student is formulated into a problem and explained in a three step processes that can be coded in a procedural programming language. The focus is on the overall design rather than how every step is conducted as each sub-function is explained in another subprogram.

The third category is object-oriented programming language such as Java and C++. A few researchers state that these languages are not suitable for teaching novices because of unnecessary complexity, inconsistency, modes and implementation issues [14]. Complexity is unnecessary if it contributes nothing to the program requirements at the moment. Inconsistency of the languages requires the students to learn the material twice. Designer mode and developer mode cause difficulties when systems did not behave as expected. Implementation issues such as efficiency issues are not to be highlighted for novice learners. The third category is suitable for intermediate and advance learners.

D. PROGRAMMING ENVIRONMENT

A few programming environments are dedicated for logic thinking using designs and others are dedicated for coding. Programming environments in this section refers to useful tools

978-1-5090-6255-3/17/$31.00 ©2017 IEEE

that are useful for programmers to develop programs from programming language code. The programming environment is categorised into three focus-based programs, which are program development, program behaviour, and programming support tool.

The first focus is on program development (i.e. visual debugger), a tool that reflects multiple code-level aspects that show execution proceeding statement by statement. The focus of these programs is line-by-line codes usually with number lines indicators to ease the writing and analysis of code based n the design of the required program. A generic platform enables teachers to create specific programming micro-worlds that match their teaching goals [16].

The second focuses on program behaviour (i.e. visualization) such as Scratch (https://scratch.mit.edu). These programs are designed for users with no or little programming experience. The aim of these programs is to provide practice in a friendly and attractive environment. Code visualization tools visualise the static structures or display the dynamic aspects of program execution, not the actual programming code [17]. The students were able to demonstrate certain computer science concepts and improve their cognitive performance achievement [18].

The third focuses on programming support tool (i.e. interactive incremental code execution). This is as extension to programming environment. This tool is found to increase understanding of object-oriented programming such as Java and Python [19].

E. Design Science Research in Information Systems

Design Science Research demands that research not only describes current situation but also produces solution that improves the current situation [20], [21]. The Information Systems discipline is in the intersection between machine knowledge and human behaviour that draws on the disciplines of natural science and social science [22].

Both behavioural science and Design Science Research paradigms are fundamental to the Information Systems discipline, as they require investigation of the people, organisational and technology aspects within a context [20]. The goal of behavioural science research is to explain or predict phenomena while Design Science Research is to provide utility for the organisational needs [23].

Design Science creates and evaluates artefacts that are broadly defined as constructs, models, methods or instantiations that can contribute to research knowledge [23], [24]. The challenge for lecturers is to change the curriculum resources, subject delivery and subject design to make best use of the resources in order to improve the students' learning. A guideline for lecturers to improve current learning and teaching programming skills is an artefact is in need.

IV. PROGRAMMING SKILLS DEVELOPMENT CONCEPTUAL FRAMEWORK

The following conceptual framework is proposed in order to analyse the programming skills development based on the research question: (1) The teaching and learning alignment as a unit of analysis (2) a holistic approach on how programming skills are developed, (3) an emphasis on activities and (4) the importance of providing guidelines for lecturers. These four dimensions are proposed for future researches on programming skills development in Information Systems.

A. TEACHING AND LEARNING ALIGNMENT AS A UNIT OF ANALYSIS

Learning plane in higher education consists of teaching and learning that requires both sides to be evaluated in order to achieve the desired learning outcome. From four learning theories namely behaviourism, cognitivism, constructivism and connectivism, constructivism provide a promising lens in evaluating teaching and learning.

Constructivism allows a learning research to focus on learners' effort to construct their own learning [9], [25]. A shift from teacher-centred to student-centred learning activities is required to enable a constructivist-learning environment as the students learn from their personal knowledge experiences [26], [27]. In higher education institutions, constructive alignment principles are used as a tool to design courses in line with outcome based education [28].

B. A HOLISTIC PERSPECTIVE ON HOW PROGRAMMING SKILLS DEVELOPED

The constructive alignment principles address three main components, which are learning objectives, learning activities and learning assessments (refer to Fig. 1). Evaluating teaching and learning planes in programming skills development provides a holistic perspective by providing insights from educators and learners. Feedback from both agents assists in improving the current situation.

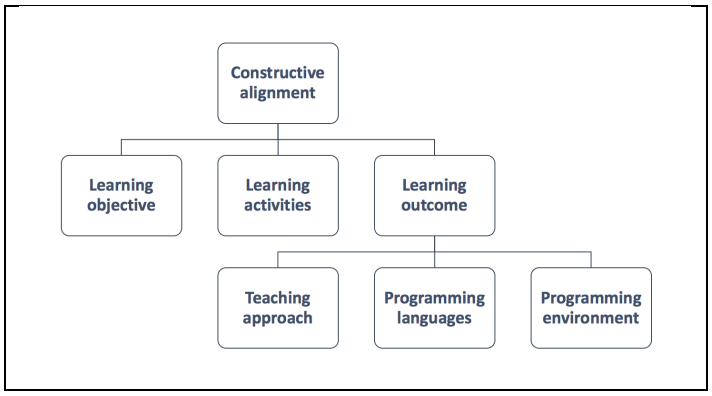

Fig. 1. Programming skills development in learning activities based on constructive alignment principles.

C. AN EMPHASIS ON LEARNING ACTIVITES

The constructivist approach to learning describes a learning process, in which students work individually or in small groups to explore, investigate and solve problems. In the traditional learning environment, the lecturer tends to emphasize on factual knowledge delivery and focus on disseminating content [29]. The learning approach tends to be passive and the students do not interact with the lecturer during the learning process. However, in a constructive learning environment, the lecturer becomes a facilitator that encourages the students to explore other materials. Students become actively engaged in seeking knowledge and information rather than operating in a passive manner [26].

978-1-5090-6255-3/17/$31.00 ©2017 IEEE

D. AN ACKNOWLEDGMENT OF THE IMPORTANCE OF PROVIDING GUIDELINES FOR LECTURERS

One of the aspects of constructive alignment is learning activities that are designed to enhance learning computer programming. In order to design learning activities, a guideline is beneficial to lecturers and course developers as an artefact.

The proposed programming skills development framework provides a guide to draw future research to produce an artefact as the outcome. A guideline for lecturers is important to gear the evaluation of the impact of the current practices with a view to provide improvement on learning.

V. CONCLUSION

This paper is towards supporting the move to change to a constructive alignment of learning computer programming. To further enhance programming skills development, continuous improvement in computer-programming courses is necessary. Brief themes of a few practical examples and suggestions were provided in this paper as guidelines for novice lecturer.

It can be concluded that additional research is needed as evidence to validate the framework and support approaches for teaching computer programming. Thus, future research is in need to understand varieties of learning and teaching programming in terms of learning objectives, activities and assessments. Further researches will provide scholarly motivations for course design choices.

In addition, this research is a platform to provide better understanding on the effective teaching and learning in a computer programming class that can affect the pedagogy in engineering and information systems.

ACKNOWLEDGMENT

This work is part of a research funded by Research Acculturation Grant Scheme from the Ministry of Higher Education, Malaysia.

REFERENCES

[1] Miliszewska, I., Venables, A., & Tan, G., "Improving progression and satisfaction rates of novice computer programming students through ACME–Analogy, Collaboration, Mentoring, and Electronic support". *Issues in Informing Science and Information Technology*, *5*, 311-323, 2008.

[2] K. Adu-Manu Sarpong, J. Kingsley Arthur, and P. Yaw Owusu Amoako, "Causes of Failure of Students in Computer Programming Courses: The Teacher – Learner Perspective," *Int. J. Comput. Appl.*, vol. 77, no. 12, pp. 27–32, 2013.

[3] Hawi, Nazir. "Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course." *Computers & Education* 54.4 (2010): 1127-1136, 2010.

[4] Mhashi, M., and A. Alakeel. "Difficulties facing students in learning computer programming skills at Tabuk University." *Proceedings of the 12th International Conference on Education and Educational Technology (EDU'13), Iwate, Japan*. 2013.

[5] Jenkins, Tony. "Teaching programming-a journey from teacher to motivator." *The 2nd Annual Conference of the LSTN Center for Information and Computer Science*. 2001.

[6] Law, Kris MY, Victor CS Lee, and Yuen-Tak Yu. "Learning motivation in e-learning facilitated computer programming courses." *Computers & Education* 55.1 (2010): 218-228, 2010.

[7] Al-Imamy, Samer, Javanshir Alizadeh, and Mohamed A. Nour. "On the development of a programming teaching tool: The effect of teaching by templates on the learning process." Journal of Information Technology Education 5.2006 (2006): 271-283, 2006.

[8] S. R. Mchugh, James A, "Methodology and technology for learning programming Article information :," *J. Syst. Inf. Technol.*, vol. 4, no. 1, pp. 23–35, 2000.

[9] E. Ultanir, "An Epistemological Glance at the Constructivist Approach: Constructivist Learning in Dewey, Piaget, and Montessori," *Int. J. Instr.*, vol. 5, no. 2, pp. 195–212, 2012.

[10] R. Horváth and S. Javorský, "New Teaching Model for Java Programming Subjects," *Procedia - Soc. Behav. Sci.*, vol. 116, pp. 5188–5193, 2014.

[11] A. Gunawardena and V. Adamchik, "A Customized Learning Objects Approach to Teaching Programming," 2000.

[12] Vihavainen, Arto, Matti Paksula, and Matti Luukkainen. "Extreme apprenticeship method in teaching programming for beginners." *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 2011.

[13] R. Shehane and S. Sherman, "Visual Teaching Model for Introducing Programming Languages," *J. Instr. Pedagog.*, vol. 14, pp. 1–8, 2014.

[14] Warren, Peter. "Teaching programming using scripting languages." *Journal of Computing Sciences in Colleges* 17.2 (2001): 205-216, 2001.

[15] M. Abdullah-Al-Wadud, "A Procedural Way of Teaching Procedural Programming Language," vol. 1, pp. 114–117, 2016.

[16] Quinson, Martin, and Gérald Oster. "A Teaching System To Learn Programming: the Programmer's Learning Machine." *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015.

[17] Zeller, Andreas, and Dorothea Lütkehaus. "DDD—a free graphical front-end for UNIX debuggers." *ACM Sigplan Notices* 31.1 (1996): 22-27.

[18] Meerbaum-Salant, Orni, and Michal Armoni. "Mordechai (Moti) Ben-Ari (2010). Learning computer science concepts with scratch." *Proceedings of the Sixth international workshop on Computing education research (ICER'10)*.

[19] Pears, Arnold N. "Enhancing student engagement in an introductory programming course." *Frontiers in Education Conference (FIE), 2010 IEEE*. IEEE, 2010.

[20] Venable, John. "The role of theory and theorising in design science research." Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006). 2006.

[21] Iivari, Juhani, and J. Venable. "Action research and design science research–seemingly similar but decisively dissimilar." *17th European conference on information systems*. Vol. 17. 2009.

[22] Gregor, Shirley. "The nature of theory in information systems." *MIS quarterly* (2006): 611-642.

[23] Von Alan, R. Hevner, et al. "Design science in information systems research." *MIS quarterly* 28.1 (2004): 75-105.

[24] March, Salvatore T., and Veda C. Storey. "Design science in the information systems discipline: an introduction to the special issue on design science research." *MIS quarterly* (2008): 725-730.

[25] M. Ben-Ari, "Constructivism in computer science education," *ACM SIGCSE Bull.*, vol. 30, no. 1, pp. 257–261, 1998.

[26] M. Tom, "Five Cs Framework: A Student-centered Approach for teaching programming courses to students with diverse disciplinary background," *J. Learn. Des.*, vol. 8, no. 1, pp. 21–37, 2015.

[27] H. Wang and A. C. L. Theory, "Learner Autonomy Based On Constructivism Learning Theory," vol. 8, no. 5, pp. 1543–1545, 2014.

[28] S. Hadjerrouit, "Object-oriented software development education: A constructivist framework," *Informatics Educ. Int. J.*, vol. 4, no. 2, pp. 167–192, 2005.

[29] Luckie, DB, Aubry, JR, Marengo, BJ, Rivkin, AM, Foos, LA & Maleszewski, JJ, 'Less teaching, more learning: 10-yr study supports increasing student learning through less coverage and more inquiry' *American Journal of Physiology - Advances in Physiology Education*, vol 36, no. 4, pp. 325-335, 2012. DOI: 10.1152/advan.00017.2012

978-1-5090-6255-3/17/$31.00 ©2017 IEEE