Development of Hardware-Interfacing Learning Kit for Novice Learning Programming

Siti Rosminah M. D. Derus and Ahmad Zamzuri Mohamad Ali

***Abstract*—Visualization tool has been proven useful in enhancing novice programmer's learning. Despite the large number of studies performed on visualization tools, it appears to be very few using it to assist students in understanding the fundamental concept of hardware software interfacing programming. Therefore, in this study, a program visualization (PV) kit was developed with the main objective is as a practical hands-on learning kit for novice in exploring and gaining basic skills of hardware software interfacing in visual environment. The ADDIE instructional design model was utilized in the design process; which consists of five principal phases namely, analysis, design, development, implementation and evaluation. The development processes was grounded on Cognitive Load Theory, Models of Human Memory and educational principles to ensure the maximum effectiveness of learning. The PV kit consist hardware visualization and software visualization. The hardware visualization is an external device that can be connected to the USB port, where the learners can visualize the input and output result from the task stimulated by the software. The software visualization contains main learning activities that have been organized from simple to complex tasks. It is expected that the PV kit could be benefited by the students in developing the basic skills of hardware software interfacing programming.**

***Index Terms*—Hardware software interfacing, instructional, programming, visualization.**

I. Introduction

The emergence of computer technology in the era of Internet and globalization has made programming essential to be learned. Nowadays, programming skills required not only in the field of Information Technology alone, but it is also needed in many other fields such as Science, Mathematics and Engineering. However, the subject is said to be difficult and complex [1]-[3], and even it is categorized as one of the seven major challenges in computing education [4].

Among the issues discussed concerning the problem faced by students, particularly novices in learning programming is difficulty in understanding the abstract concepts and programming development environment and correlate it with actual problems. Notably, novice knowledge more likely to be context-specific and fragile, and they also frequently find it difficult to apply it properly [5]. They know the syntax and semantics of the specific programming statements, but they don't know how to incorporate these features into the program [6]. This factor is due to the limited surface and superficial of knowledge, so they can't build mental models perfectly in solving the problems involves programming development process [7], [8]. Besides that, students are not capable in visualizing the programming flow which appears to be among the factors that attribute to difficulty in obtaining programming skills [9], [10]. They face difficulty in visualizing the processes in computer memory when the program executed, and this will attribute to failure in constructing a viable mental model of overall programming concept.

To address the problem stated above, effective instructional strategies with the aid of appropriate learning tools, need to be provided to ensure optimum learning outcomes. Program Visualization (PV) is a learning support tool, which can help in strengthen the understanding of the student in comprehend the process of programming [5], [11]-[13]. This is moreover true, since humans have better ability in processing visual information [14]. Thus, PV has promising potential in assisting students to understand the key element relating to the program code execution through a graphical display in a more dynamic presentation.

Manuscript received November 12, 2014; revised March 4, 2015.

Siti Rosminah M. D. Derus and Ahmad Zamzuri Mohamad Ali are with Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Tanjong Malim, 35900, Perak, Malaysia (e-mail: zamzuri@fskik.upsi.edu.my).

II. Research Background

Programming involves many dynamic concepts of mental grasp, causing students' difficulty in understanding it through the use of static media such as textbooks, projected presentation, whiteboard and verbal explanation [15]. Therefore, Program Visualization (PV) provides a dynamic environment that represent programming concepts for a better understanding [16]. PV can be defined as an approach or technique in the graphical displays that shows the changes in the program code execution [17]. The main goal of PV is to assist students in understanding the dynamic behavior of the program by displaying aspects like values of variables, evaluation of statements and changes in the program state in general. In addition, PV could also explain visually the hidden processes during the program run-time [18]. Generally, PV consists of two categories: dynamic PV and static PV [19]. Dynamic PV visualizes the flow of programming execution of each line of code including the changes of the variables value. Static PV visualizes program structures and relations between program objects.

There are many PV tools developed for learning programming subject. JavaVis [20] is a PV tool developed to visualize the execution of program code through the object and sequence diagram. JIVE [21] has a characteristic of code highlighting and visualizes the object structure and the calling sequence of methods. BlueJ [22] is a static PV tool has a class view showing relations between classes and an object dock containing all initialized objects. Jeliot3 [23] illustrates semi-automatic visualization of the data and

control flows. ViLLE [24] provides parallel view displaying a program in two languages simultaneously (Java and C++), and its built-in editor for interactive quizzes and tests displayed as pop-up windows. Frances-A [25] have the ability to represent the programming code between high-level code and machine code, also capable of illustrate machine state transition. 3De [26] represent multi abstract view of the programming stages starting designing problem-solving, developing code and validating logical flow.

To date, there is a deficiency PV developed to assist students in understanding the fundamental concept of hardware-software interfacing programming. Hardware-software interfacing programming is the concept where programs code are developed to interface with real world devices [27]. The program code will communicate with a hardware interface via the port to enable the external device automatically to perform the task needed. A port is an electronic hardware that is used as an interface to connect with another electronic device for the purpose of information exchange [28]. Most of the visualization tools for the purpose of learning the fundamental concept of hardware-software interfacing available in the market were merely microcontroller technology oriented, which only visualize the hardware execution.

III. Research Objectives

Most of the visualization tool in the market unable to fulfill the need of hardware-software interfacing programming learning, specifically for novices. Therefore, the objective of this study is to develop a program visualization kit to assist students, especially novice in understanding the basic concepts of hardware-software interfacing programming.

IV. Research Methodology

PV design and development process is based on the principles of instructional design framework. The selection of the appropriate model for instructional design is very important in providing a framework that can help a developer to implement tasks in design and develop learning support tools in a more systematic manner. In this study, ADDIE development model was chosen because it is a commonly used model in the development of computer technology-based learning media [29], [30]. The ADDIE model includes five phases: 1) analysis, 2) design, 3) development, 4) implementation and v) evaluation.

A. Analysis Phase

Analysis phase is the first stage in the process of development of PV where it consists of three main activities: 1) need assessment, 2) setting the goal, and 3) identify the students' background.

Reference [31] defines, "needs is the gap between what is expected and the existing conditions". Consequently, based on interviews with two polytechnic lecturers who teach Hardware Interfacing course, found that they face difficulty in teaching interfacing programming topics, as there is no suitable learning aids to assist students in understanding the concepts of how to manipulate the input-output data process between external device and computer systems. In addition, the students also failed to obtain a clear picture of the execution of the program code through the slide presentation only. Therefore, it is crucial to develop a visualization tool that can facilitate them in understanding the fundamental concept of hardware interfacing.

Having identified the problems, the researcher set a goal of development the visualization kit is to be used as a practical hands-on learning kit to assist students to explore and master the hardware-software interfacing fundamental concept in visual environment. However, to ensure the goal can be achieved, it is vital to gather the information about the students' background and their prior knowledge. Therefore, the minimum prior knowledge required when using PV kit is understand the basic principles of digital logic and the ability to change the number system. This is because interfacing programming often involves a number system. The computer consists of the components of the electricity generated by electric power only works in two conditions, whether there is a current flow (on) or no current flow (off) [32]. These conditions represented by the numbers '1' for current flow and '0' for no current flow, and this number system is called binary number system. However, humans have difficulties if the binary number involving the use of large amounts of data. Thus, the hexadecimal number system is used to overcome this problem, which consists of 16 unique symbols: the numbers 0 to 9 and the letters A to F. In addition to affecting the number system, the other matter had to be understood as the concept of the hardware-software interface is how the two different parts can interact to perform the required functions through the communication port. To do this, we need to send the electrical signal by sending a byte of information to the port. A byte is a unit of data that is eight binary digits long, and most computers use to represent a character such a letter, number or typographic symbol [27].

B. Design Phase

The design phase is the process of transferring information from the analysis phase to a physical sketch that will be used during the development process. Fig. 1 shows the design process of the visualization kit.

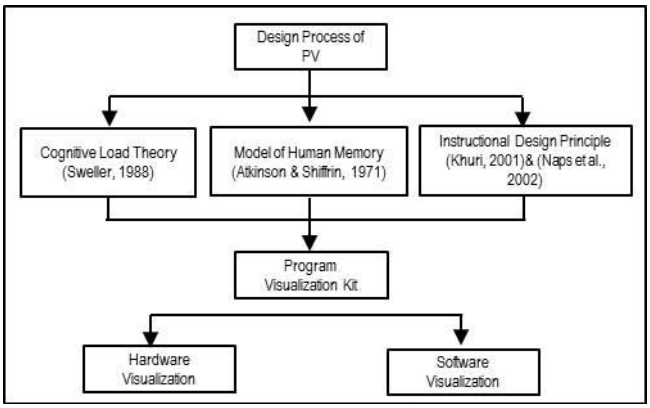

Fig. 1. Design process of PV.

PV design process taking into account the approach of Cognitive Load Theory [33] and Models of Human Memory [34]. Cognitive Load Theory and Models of Human Memory are closely linked to raising the level of cognitive involving working memory and long term memory. Due to the limited capacity of working memory which is 15 seconds to 30 seconds [35], the visual information presented should be

integrated with the scheme in the preexisting long-term memory to complete the development of mental models in working memory and reduce the cognitive load.

In order to expedite formation of a mental model that visual information transmitted can provide meaningful information, PV architecture is designed in two forms which consist of the hardware visualization and the software visualization. Hardware visualization is an external circuit connected to the USB port on the computer system, in which students will be able to view the real-world application of input and output data manipulation. In addition, these tools can be used to configure, control, and connect as external devices with Visual software environment to enable students gain real experience to visualize the effect of program code have been created. Whereas the software visualization is a graphical interface that allows students to interact visually on the monitor screen and contains main learning activities. Hence, the content of learning activities will be organized from simple to complex tasks to facilitate students to easily store the information received into long-term memory for schema formation and thus reduce cognitive load [36]. In addition, the principles in terms of the display layout, use of color and interactivity issues are also taken into account as proposed by [37] and also [38].

C. Development Phase

The third phase is the development of which involves the process of 'translating' activities determined in the analysis and design phase. As notified through the design phase, PV kit will be developed in two forms, namely hardware visualization and software visualization.

The hardware visualization development begins with identifying the interface between the external devices to the computer system. For this study, the USB (Universal Serial Bus) port is used as an interface because it is the most common type of computer port used in today's computers. The USB port send data in series, which is mean sending one bit of data at a time but its data transfer speed is high, either 1.5 Megabits per second or 12 Megabits per second over four-wire cable [28]. However, we need a parallel communication as an output to view the 8 bit LED to show effect of the manipulation output data when program code are executed (Fig. 2). Parallel communication is a method of conveying multiple binary digits (bits) simultaneously. Therefore, the Universal Asynchronous Receiver/Transmitter (UART) is used to make sure the output will be manipulated simultaneously. UART is a device to convert the data series to parallel form and vice versa.

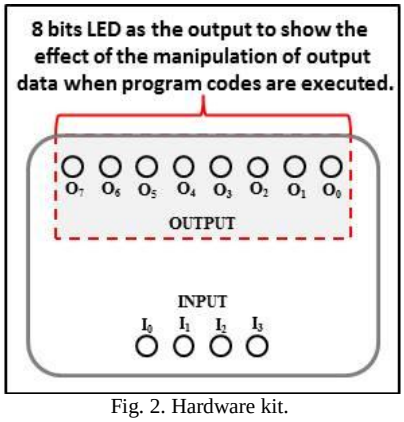

Fig. 2. Hardware kit.

For operational control of the hardware, micro controller is used. Micro controller is a small computer on a single integrated circuit containing all the major components such as CPU, RAM, ROM, input/output port and serial communication interface in one chip.

The software-development process uses Visual software for the development of the graphical interface that contains a variety of learning activities. Taking into consideration the limitations of working memory, this graphics interface is divided into several segments and organized into five separate tabs display (Fig. 3) : 1) Introduction tab, 2) Decimal tab, 3) Hexadecimal tab, 4) Practice tab and v) Interfaces tab. Introduction tab brief a description of the basic concepts of hardware and software interfacing. Decimal tab contains activities to configure the input and output data using a decimal numbering system, while the hexadecimal tab contains activities to configure the input and output data using a hexadecimal numbering system.

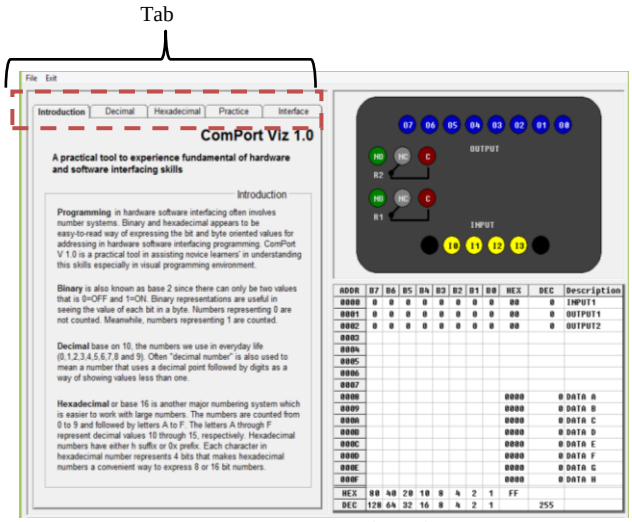

Fig. 3. Graphical interface of PV.

Practice tab contains practical activities to understand the concept of interfacing between the external devices with programming code provided. This section includes a worked example of program code where students can visually look on the changes in every line of program executed. According to [39] students often looking for worked example rather than text explanation as the primary and most commonly source in the learning materials. Past research has shown that examples play an important role in learning [40], [41] and are vital to the acquisition of initial cognitive skills [42], [43]. Matrix table shows the contents of data stored in specific address locations and the changes of variable value to help students understand what happens to this data during the program code execution. Therefore, it guide to student familiarize the role of variable position in the computer memory when program code executed [44]. Furthermore, with the combination of output view and matrix table, the redundancy effect has been discovered as proposed by cognitive load theory. The redundancy effect occurs when the information are presented in multiple sources without reference to each other [39].

The interface tab briefly explains how to use the PV kit in VB environment. Therefore, the student will have better experience in manipulating input-output data process via VB environment and use the PV kit as the external device and

interfaced it.

D. Implementation and Evaluation Phase

Usability study at one of the polytechnic in Malaysia was conducted once the PV kit has been developed completely. A total of 25 students enrolled Diploma in Computer Technology involved in the study. These students have been selected based on their basic knowledge of VB programming, hardware and software interfaces and electronic engineering fundamentals. The aim was to test the effectiveness of the VP kit and identifying the problems that were not identified throughout the design and development phases. In addition, it aims to get students' views on the appropriateness of the VP kits to help them understand the basic concepts of hardware-software interfacing programming. Overall, the majority of students indicate they strongly agree and are satisfied with the VP kit developed and agree it is appropriate for the use in learning software programming and hardware interfacing.

V. Conclusion

It is undeniable programming is a very challenging subject to learn because it involves understanding the abstract concept. Combination of effective learning strategies and the use of appropriate learning materials have the potential to address particular problems in learning programming. Therefore, this study intended to design and develop a learning kit based on the program visualization concept to facilitate students learn programming especially in understanding the basic concepts of hardware-software interfacing. The design and development processes were grounded on ADDIE instructional design model. In the process of development, methods of presenting learning information are vital to ensure for better learning understanding and performance. Therefore, Cognitive Load Theory, Models of Human Memory and educational principles in terms of layout, use of color and interactivity issues also were taken into consideration, to ensure the maximum effectiveness of learning. It is expected that the development of PV kits can be utilized by students to acquire programming skills, especially the concept of hardware-software interfacing.

References

[1] T. Jenkins, "On the difficulty of learning to program," in *Proc. the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002, vol. 4, pp. 53-58.

[2] I. T. C. Mow, "Issues and difficulties in teaching novice computer programming," in *Innovative Techniques Technology, e-Learning, e-Assessment and Education*, Magued Iskander, ed., Springer, 2008, pp. 199-204.

[3] M. Yousoof, M. Sapiyan, and K. Kamaluddin, "Reducing cognitive load in learning computer programming," in *Proc. the World Aademy of Sciene, Engineering and Technology*, 2005, vol. 1, no. 12, pp. 469-472.

[4] A. McGetrick, R. Borle, R. Ibbett, J. Llyod, G. Lovegrove, and K. Mander, "Grand challenges in computing: Education - A summary," *The Computer Journal*, vol. 48, no. 1, pp. 42-48, 2005.

[5] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 14, 2005.

[6] L. E. Winslow, "Programming pedagogy - A psychological overview," *ACM SIGCSE Bulletin*, vol. 28, no. 3, pp. 17-22, 1996.

[7] M. Reginamary, S. H. Hew, and A. C. Koo, "Multimedia learning object to build cognitive understanding in learning introductory programming," in *Proc. the 7th International Conference on Advances in Mobile Computing and Multimedia -MoMM '09*, 2009, pp. 396-400.

[8] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137-172, 2003.

[9] M. A. Rizal, M. Nurliana, and B. Yahya, "The effect of using a learning model based on problem solving method on students with different cognitive style and logic ability," presented at 2nd International Malaysian Educational Technology Convention, 2007.

[10] I. Milne and I. Rowe, "Difficulties in learning and teaching trogramming: Views of students and tutors," *Education and Information Technologies*, vol. 7, no. 1, pp. 55-66, 2002.

[11] M. D. S. Rosminah and M. A. A. Zamzuri, "Integration of visualizaiton techniques and active learning strategy in learning computer programming: A proposed framework," *International Journal on New Trends in Education and Their Implications*, vol. 5, no. 1, pp. 93-103, 2014.

[12] T. Linden and R. Lederman, "Creating visualizations from multimedia building blocks: A simple approach to teaching programming concepts," in *Proc. Information Systems Educators Conference*, Wilmington North Carolina, 2011, pp. 1-10.

[13] J. Helminen and L. Malmi, "Jype – A program visualization and programming exercise tool for python categories and subject descriptors," in *Proc. the 5th International Symposium on Software Visualization*, 2010, pp. 153–162.

[14] A. Pears, S. Seidman, L. Malmi *et al*., "A survey of literature on the teaching of introductory programming," in *Proc. ITiCSE-WGR '07 Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, 2007, pp. 204-223.

[15] E. Fouh, M. Akbar, and C. A. Shaffer, "The role of visualization in computer science education," *Computers in the Schools*, vol. 29, no. 1-2, pp. 95-117, 2012.

[16] A. Gomes and A. J. Mendes, "An environment to improve programming education," in *Proc. the 2007 International Conference on Computer Systems and Technologies - CompSysTech '07*, New York, USA: ACM Press, 2007, p. 1.

[17] D. Gračanin, K. Matković, and M. Eltoweissy, "Software visualization," *Innovations in Systems and Software Engineering*, vol. 1, no. 2, pp. 221-230, 2005.

[18] R. Bednarik, A. Moreno, N. Myller, and E. Sutinen, "Smart program visualization technologies: Planning a next step," in *Proc. Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 2005, pp. 717-721.

[19] T. Rajala, M. -J. Laaksi, E. Kaila, and T. Salakoski, "Effectiveness of program visualization: A case study with the ViLLE tool," *Journal of Information Technology Education: Innovatins in Practice*, vol. 7, pp. 15-32, 2008.

[20] R. Oechsle and T. Schmitt, "Javavis: Automatic program visualization with object and sequence diagrams using the java debug interface (jdi)," *Lecture Notes in Software Visualization*, vol. 2269, pp. 176-190, 2002.

[21] P. Gestwicki and B. Jayaraman, "Interactive visualization of java programs," in *Proc. IEEE Symposia on Human Centric Computing Languages and Environments*, 2002, pp. 226-235.

[22] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg, "The BlueJ system and its pedagogy," *Journal of Computer Science Education*, vol. 13, no. 4, pp. 1-12, 2003.

[23] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, "Visualizing programs with Jeliot 3," in *Proc. the Working Conference on Advanced Visual Interfaces - AVI '04*, 2004, p. 373.

[24] T. Rajala, M.-J. Laakso, E. Kaila, and T. Salakoski. (2008). VILLE - A language-independent program visualization tool. [Online]. Available: http://verkkoopetus.cs.utu.fi/koli/2008/ville/ViLLE_system.pdf

[25] T. Sondag, K. L. Pokorny, and H. Rajan, "FRANCES-A: A tool for architecture level program visualization," *J. Comput. Small Coll.*, pp. 283-292, 2011.

[26] Affandy, N. Suryana *et al*., "3de - Synergetic program visualization: A visual learning-aid tool for novice students," in *Proc. 2011 International Conference on e-Education, Entertainment and e-Management (ICEEE)*, 2011, pp. 133-137.

[27] J. Katupitiya and K. Bentley, *Interfacing with C++*, Berlin, Heidelberg: Springer, 2006.

[28] J. Axelson, *Parallel Port Complete: Programming, Interfacing & Using the PC's Parallel Printer Port*, Lakeview Research, 2000.

[29] H. Jamaluddin, A. Baharuddin, and T. Zaidatun, *Pembangunan Perisian Multimedia: Satu Pendekatan Sistematik (Multimedia Development: A Systematic Approach)*, Kuala Lumpur: Venton Publishing, 2001, pp. 29-31.

[30] S. Rio Sumarni, "Design of instructional materials for teaching and learning purpses: Theory into practice," *Malaysian Education Deans' Council Journal*, vol. 1, pp. 97-110, 2007.

[31] G. R. Morrison, S. M. Ross, and J. E. Kemp, *Designing Effective Instruction*, 4th ed., John Wiley & Sons, Inc, 2004, p. 32.
[32] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann, 2005.
[33] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive Science*, vol. 12, no. 2, pp. 257-285, 1998.
[34] R. C. Atkinson and R. M. Shiffrin, "The control of short term memory," *Scientific American*, vol. 225, issue 2, pp. 82-90, 1971.
[35] E. B. Goldstein, *Cognitive Psychology: Connecting Mind, Research, and Everyday Experience*, 3rd Ed., Belmont, CA: Wadsworth Cengage Learning, 2011.
[36] J. J. G. V. Merri ënboer and P. Ayres, "Research on cognitive load theory and its design implications for e-learning," *Educational Technology Research and Development*, vol. 53, no. 3, pp. 5-13, 2005.
[37] S. Khuri, "A user-centred approach for designing algorithm visualizations," *Informatik/Informatique, Special Issue on Visualization of Software*, vol. 2, no. 2, pp. 12-16, 2001.
[38] T. L. Naps, G. Rößling, Almstrum *et al*., "Exploring the role of visualization and engagement in computer science education," in *Proc. ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*, 2002, pp. 131-152.
[39] J. Sweller, J. J. G. V. Merrienboer, and F. G. W. C. Paas, "Cognitive architecture and instructional design," *Education Psychology Review*, vol. 10, no. 3, pp. 251-296, 1998.
[40] M. Caspersen and J. Bennedsen, "Instructional design of a programming course: A learning theoretic approach," in *Proc. The Third International Workshop on Computing Education Research*, 2007, pp. 111-122.
[41] M. T. H. Chi, M. Bassok, M. W. Lewis, P. Reimann, and R. Glaser, "How students study and use examples in learning to solve problems," *Cognitive Science*, vol. 13, pp. 145-182, 1989.
[42] S. S. A. Rahman and B. D. Boulay, "Schema aquisition: Implications for the instructional design of examples," in *Proc. Conference on Artificial Intelligence in Education*, 2009, pp. 757-758.
[43] R. K. Atkinson, S. J. Derry, A. Renkl, and D. Wortham, "Learning from examples: Instructional principles from the worked examples research," *Review of Educational Research*, vol. 70, pp. 181-214, 2000.
[44] M. D. S. Rosminah and M. A. A. Zamzuri, "Difficulties in learning programming: Views of students," in *Proc. 1st International Conference on Current Issues in Education*, Yogyakarta: Yogyakarta State University, 2012, pp. 74-78.

Siti Rosminah M. D. Derus is currently a PhD student in the Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Malaysia. She has a bachelor degree in electrical engineering (computer technology) from Universiti Teknologi Malaysia, and a master in education from Universiti Teknologi Malaysia. Her research interests are instructional technology and programming visualization.

Ahmad Zamzuri Mohamad Ali is an associate professor of multimedia in the Faculty of Art, Computing and Creative Industry, Universiti endidikan Sultan Idris, Malaysia. He has a bachelor degree in electrical engineering from Universiti Teknologi Malaysia, a master in education from Universiti Teknologi Malaysia and a PhD in multimedia design from Universiti Sains Malaysia. He has taught both face-to-face and online classes in higher education for over 15 years. His research and publication interests are multimedia design, instructional technology, ICT in education and open source in education.