International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5599~5606
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5599-5606

Analysis of the learning object-oriented programming factors

Qais Ali Batiha, Nazatul Aini Abd Majid, Noraidah Sahari, Noorazean Mohd Ali
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia

Article Info

Article history:

Received Jan 24, 2023
Revised Mar 11, 2023
Accepted Mar 28, 2023

Keywords:

Effectiveness learning
Learning environment
Learning object-oriented factors
Motivation
Object-oriented programming difficulties

ABSTRACT

Students often feel overwhelmed by object-oriented programming courses. They find it difficult and complex to learn, requiring a high cognitive load to use the concepts in coding. These issues lead to demotivation in learning programming. This research aims to identify and verify factors that contribute to learning object-oriented programming from two perspectives: interviews and surveys. A literature review was conducted to identify these factors, followed by interviews with five experts who have been teaching object-oriented programming for over ten years to confirm them. Based on the interview results, a questionnaire was developed and administered to 31 bachelor students and 19 lecturers with master's or doctorate degrees in computer science. The responses indicated that the identified factors were acceptable, with scores ranging from 3.74 to 4.65. The outcomes of this study are a set of factors that should be considered in a programming environment to improve the teaching and learning of object-oriented programming and make it more accessible and engaging for students.

Corresponding Author:

Nazatul Aini Abd Majid
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia
Bangi, 43600, Malaysia
Email: nazatulaini@ukm.edu.my

1. INTRODUCTION

Research shows that it is difficult for students to learn object-oriented programming languages like Java, Python, or C++ because they are based on abstract concepts like class, inheritance, and polymorphism [1]–[4]. Factors such as problem complexity, programming environment, and students' skills can also make learning difficult. To improve learning, integrated approaches such as hands-on learning or visualized programming environments have been developed. Examples of these visualized programming environments include BlueJ [5], Greenfoot [6], and object-oriented puzzle programming (OOPP) [7].

According to Su and Hsu [8], reducing students' difficulties in grasping the concepts, principles, and rules of an object-oriented programming language may improve their performance, competency, and motivation to learn the language. As a result, various studies have been conducted on making programming more motivating and interesting, enhancing performance, and boosting programmers' confidence [9], [10]. Therefore, in order for students to learn an object-oriented programming language, it is necessary to determine the fundamental factors that affect their learning. The goal of this work is to develop a list of programming learning factors that can help students learn an object-oriented programming language. This list of factors will then be validated using qualitative and quantitative methods from different perspectives.

In the first section of this study, we provide a concise explanation of the study's main topic, as well as the goals and how they are accomplished. The challenges and issues are summarized in section 2, which builds on prior work. In section 3, relevant works are summarized and critically evaluated. The methodologies used in this investigation are discussed in section 4. Section 5 presents the analysis of the results and discusses the findings based on various perspectives. Finally, section 6 presents the conclusion of this research.

2. PROGRAMMING LANGUAGE ISSUES AND CHALLENGES

According to Robins [11], students often encounter challenges in the form of success/failure rates as well as the complexity of programming (such as the code, the object, the model, and the display) when enrolling in programming courses. However, there are effective and innovative approaches to teaching programming skills that motivate novice students to learn programming while also adding value to the teaching process. There are many learning tools for graphical programming, which is more appealing than textual programming. As college students are at an age where they can start learning how to program and solve problems, these tools have been proposed as part of university learning methods [12].

In general, logic is the most crucial aspect of the programming process, and it is also the first step in the creation of a complete program. Many high-level languages, such as Java, C++, and Arduino, have similar syntactic and semantic rules. However, writing programs in a specific programming language requires that the programmer be familiar with the language's structure and be able to recall certain concepts. While the complex syntax of programming languages may be easy for experts to understand, it offers no instructional benefit to beginners. As a result, many students struggle to grasp basic concepts, such as object-oriented management or the development of an algorithm to solve a problem.

The challenges that students face when learning object-oriented programming have led to the development of various educational programming environments in recent years. The design of these environments is a crucial topic as it can impact the success of students' learning. Hence, this research aims to explore the factors that educational programming environments possess to enhance students' learning outcomes and identify the main factors that should be considered in any learning environment for object-oriented programming.

Mackin [13] claims that the turtle graphics library is included in the logo programming language. Papert's theory on mathematics teaching inspired this logo-based effort, so Turtle graphics are often used in introductory computer science courses. The process of sketching on paper with a pen served as a point of comparison for the physical and graphical models that were the basis for the design of the turtle graphics, making the concept easy for students to understand.

Alice is a block-based narrative environment designed to help at-risk female students increase their chances of success in computer science 1 and lower-level courses [14], [15]. An add-on for Alice, called AliCe-ViLlagE, was released in 2014 with the aim of improving students' confidence and collaboration by integrating the Alice environment with a pair of programming methods [16]. With Alice and AliCe-ViLlagE, students can easily observe the behavior of their animated programs. The visual feedback provided allows students to make connections between the program "pieces" and the actions they observe in the animations. However, while Al-Jarrah and Pontelli [16] have found that this tool has a positive effect compared to Alice, there is still a lack of evaluation of its effectiveness for teaching object-oriented programming to university students. Additionally, some learners have expressed a preference for seeing the real Java code as this would help them become more familiar with industry-standard coding practices [17]. It is worth noting that Alice only provides an object representation of the program, which excludes learning the programming language.

The Greenfoot system is an educational development tool designed to teach programming to pre-university students [6]. Greenfoot combines Java programming with interactive and graphical outputs as a standard. It encourages learners to experiment with the appearance and behavior of objects using the Greenfoot interface, which allows them to create new functions or modify Java code using the built-in editor. However, the strict syntax rules of the Java language can be challenging for inexperienced learners, who may have difficulty detecting and correcting syntax errors in their code. By providing a more interactive and user-friendly learning environment, Greenfoot enables students to develop their programming skills and gain a deeper understanding of object-oriented concepts.

To investigate the use of block programming on touchscreen devices, a co-located collaborative block-based programming environment called multi-device grace was developed [18]. This environment allows multiple users to work on the same program on different devices, enabling them to collaborate and share their code in real-time. While the environment was designed specifically for touch-enabled devices, there is a lack of research on the effectiveness of using it to teach object-oriented concepts in education.

Snap! is a visual programming language designed for children, high school, and university students. It is based on a system of blocks, which makes it easy to use and understand [19]. The language allows users to define new blocks, which helps to extend its capabilities and makes it more versatile. The Snap! extension, NetsBlox [20], adds networking elements to the visual programming paradigm, enabling students to create distributed applications and collaborate on projects in a Google Docs-like environment. However, there is a lack of research on its effectiveness with object-oriented concepts, making it unclear if Snap! can be successfully used for object-oriented programming. Further research is needed to address this gap in understanding.

The OOPP environment was developed as a way to support object-oriented programming, a popular programming paradigm that was first presented by Ferrari *et al.* [7]. The environment is specifically designed

to help school students and students taking computer science 1 courses learn how to program. However, no research has been done on the usefulness and effectiveness of object-oriented blocks for computer science 2 students. This lack of research makes it difficult to determine the extent to which these blocks can help students in programming courses.

Learning an object-oriented programming language can be difficult for inexperienced students. In order to make programming more accessible to these students, some educators have turned to visual programming languages like visual logic or to more beginner-friendly languages like Python to simplify programming and make it easier for students to understand [21]. However, while Python may be suitable for introducing students to the basics of computer science, it may not be the best choice for learning higher-level concepts like object-oriented programming [22]–[24]. In order to effectively teach these more advanced concepts, educators may need to use different approaches. It is important to carefully consider the needs and abilities of students when choosing a programming language or other learning tools.

3. FACTORS TO ENHANCE THE LEARNING OF OBJECT-ORIENTED PROGRAMMING

This section provides a detailed explanation of the various learning needs for an object-oriented programming language and explores effective strategies for fulfilling these needs. The research that has been conducted in the past forms the foundation for this section, and the learning factors for an object-oriented programming language are addressed implicitly in the research investigations. The following section elucidates the learning factors that are implicit in the research inquiries.

McNerney [25], [26] proposed that easy debugging capabilities should be included in learning programming to assist novices. Papert [27] also suggested that the use of mathematical or geometric operations can help students become proficient in a particular programming language. On the other hand, [28], [29] argued that simplicity and difficulty level are important for learning a programming language. Simplicity refers to the ease of understanding and usage and should involve limiting the number of command instructions. A simple language can be acquired quickly, giving students more time to apply the language they are learning to the content of their courses. Csikszentmihalyi [30] after students are familiar with event handling and function call blocks, they should be introduced to additional complexity and abstraction, as it requires a significant degree of abstract thinking, logic, and conventional programming skills.

On the other hand, DeRose and Laurel [31] suggested that individuals who want to learn programming should engage in group interaction and utilize modularity in specific environments to enhance their understanding of the complex programming process. Horn and Jacob emphasized the importance of reality-based interaction in programming languages, arguing that it should take place in real life in order to engage learners in a more realistic manner [32]. McNerney [25] also argued that collaborative programming should be used to help novices learn more, achieve statistically significant progress, and increase students' interest in learning. Latih *et al.* [33] emphasized the importance of practical experience in acquiring programming development abilities, stating that students must engage in regular practice in order to effectively develop and improve their programming skills. They suggested that a strong practical ability is essential for success in programming and should be a key focus for students learning the subject. To conclude, Table 1 summarizes the object-oriented programming learning factors based on previous research.

Table 1. Learning factors based on previous research

No	Programming learning factors	References
1	Easy debugging (e.g., Connecting between blocks, messages, or executing the code to see the result)	[27], [29]
2	Specific operations	[27]
3	Collaborative programming	[25]
4	Simplicity	[28], [29]
5	Difficulty level	[30]
6	Practices	[30], [33]
7	Collaborative learning.	[31]
8	Object representation (Microworld)	[32]

4. METHOD

This research was conducted using two types of research methods: quantitative and qualitative. Firstly, the literature on learning programming/object-oriented programming was reviewed with the aim of identifying the main learning constituents. Then, the research employed interviews designed to identify the experts' perspectives in the context described at the beginning of the research. Next, a questionnaire instrument was used as part of a survey method to determine the opinions of the actual users who had taken the object-oriented programming course, as shown in Figure 1. The research methods used are explained in detail in the next section.

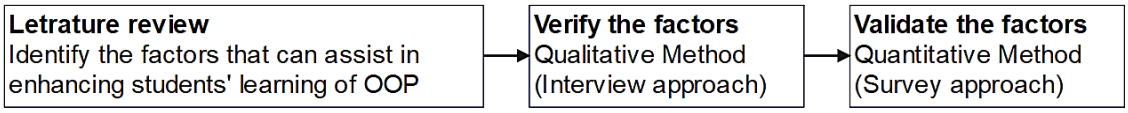

Figure 1. Research method

The goal of the interview was to confirm the factors involved in the process of acquiring skills in object-oriented programming, which had already been explored in previous research. The investigation was carried out by means of a qualitative technique, through an interview-based approach, at the Faculty of Information Science and Technology (SoftAM) located at the Universiti Kebangsaan Malaysia (UKM). An explanation of the research's goal and the rationale behind using the interview method was provided at the beginning of the interview. Based on [34], a total of five respondents were interviewed, each of whom had been teaching object-oriented programming for more than ten years. All questions related to each element were asked during the 1-2 hour-long interviews.

The research used a quantitative approach based on a survey method to gather data. The questionnaire was distributed to a sample of 50 individuals who had prior experience with object-oriented programming, including 31 students and 19 lecturers from Jordan University of Science and Technology and Irbid National University, with ages ranging from 19 to 45 years. The questionnaire was designed to gather feedback on the identified factors from actual users and included an explanation of the purpose of the research and the rationale for choosing a questionnaire approach.

The questions in the survey were carefully formulated based on previous research and validated through interviews with five experts in the field of object-oriented programming [34]. The aim was to gather valuable insights and recommendations on how to improve the learning of object-oriented programming for students. Respondents were asked to consider the various factors that influence the learning process, which were identified through previous research and further validated through interviews with experts, as referenced in [29], [35], [36]. Each respondent was asked to provide their own recommendations on how these factors could be leveraged to enhance student learning and success in object-oriented programming.

4.1. Data collection

The interviews were carried out face-to-face at UKM, and the interview questions were adopted from previous research and are listed in Table 2. For the survey, a Likert scale questionnaire was distributed to students for self-administration. The survey questions were based on the results of the interviews, and the purpose was to compare the opinions of experts and students from different generations on the same factors.

Table 2. Questions for interview

No.	Questions
1	What should be the learning factors for object-oriented programming and why?
2	Do you think these factors can support students in learning object-oriented programming? Easy debugging (Response time to events, Real-time reaction); Simplicity; Power (How easy it is to use a tool for complex problems); Practices; Difficulty level; Pair programming; Object representation; Specific operation;
3	What do you think about how these factors affect learning object-oriented programming?
4	In your opinion, how might these factors increase students' motivation and effectiveness while studying an object-oriented programming language?
5	In your opinion, do you agree that these materials would help to learn object-oriented programming? Interactive visualizations tool. Interactive environment. Lecture notes/copies of transparencies. Programming course book.
6	In your opinion, do you agree that any of these situations would help to learn object-oriented programming more effectively? In practical. Consultation or discussion with lecturers, tutors, seniors or friends In group exercise sessions While working alone on programming coursework

4.2. Survey instrument

The instrument used in this research employed Likert scales with five points to measure the level of agreement, with options ranging from strongly disagree to strongly agree. Students were asked to select one scale that best reflected their beliefs for each question. The survey questions were developed based on prior research and the interview schedule. The survey aimed to measure the level of agreement among actual users.

5. RESULTS AND DISCUSSION

The data collection techniques yielded some significant results. A list of factors for the programming learning environment was generated as a result of these findings. These factors are very important as they determine what should be developed and designed, as well as how the environment should be developed and designed. Therefore, both the results and discussions of the approaches (interview and survey) are provided in detail in this section.

5.1. Interview approach

The interview results showed that all participants agreed that the identified factors were important for keeping students motivated in learning object-oriented programming. These factors included the use of visualization or interactive environment to enhance student learning, as well as the importance of practice, consultation with lecturers, tutors, seniors, or friends, and group sessions for helping students learn object-oriented programming. The participants also concluded that these factors were important for creating a positive learning environment and promoting student success in learning object-oriented programming. Overall, the participants emphasized the importance of creating a supportive and interactive learning environment, as well as providing opportunities for practice, consultation, and collaboration, in order to help students learn object-oriented programming effectively and build their programming skills. The following are a few examples from the interviews that highlight the importance of the identified factors in keeping students motivated to learn object-oriented programming:

> "*The essential factor is to improve their motivation using any technique such as working in groups, a discussion between students, robots, games, practices, or challenging them.*" (Interviewee1).

> "*Most of the factors are enough to enhance students' learning and pair programming, and it will enhance student effectiveness and motivation.*" (Interviewee2).

> "*Using the Games/Visualization tools to motivate students in programming for a certain period and then moving to textual programming is a good idea. The reason is that students have problems with syntax errors, and it makes them more confident.*" (Interviewee3).

Based on the above, all the interviewees emphasized the importance of different methods or tools that would facilitate the learning process of object-oriented programming, increase motivation, and make programming more understandable for the students. To confirm the evaluation indicated above, the students' and lecturers' views on these factors were evaluated. The questionnaire was used as a research instrument, as shown in the next section.

5.2. Survey approach

Based on the categories shown in Table 3, the mean level is considered "high". Based on Table 4, the minimum value of the mean from the instructor's perspective is 4.10 for the specific operations factor, and from the perspective of the undergraduate students, it is 3.74 for the pair programming factor. This demonstrates that respondents agree with all the programming factors. The first prerequisite, "easy debugging", gets a rating of 4.60 from the perspective of instructors and a rating of 3.77 from the perspective of undergraduates. However, for "motivating students via the use of block-based collaborative technology," the score is 4.45 from the instructors' perspective and 4.16 from the perspective of undergraduate students. To summarize, the results were satisfactory to both instructors and undergraduates. However, the instructors' results outperformed the undergraduates for a variety of reasons, the most important of which is that the instructors are highly experienced in object-oriented programming materials and are aware of the most critical factors for student learning, mainly because most instructors have taught this course for more than five years.

Table 3. Likert classification based on [37]

Mean scores	Categorized
1.0-2.33	Low
2.34-3.67	Moderate
3.68-5	High

Table 4. The descriptive result and correlation between factors and motivation of the survey instrument

Programming Learning Factors	Kendall tau		Students (N=31)		Experts (N=19)	
	tau-b	*p*	μ	Level	μ	Level
Easy debugging	0.419**	0.000	3.77	High	4.60	High
Object representation	0.396*	0.013	3.94	High	4.16	High
Simplicity	0.450**	0.006	4.16	High	4.60	High
Difficulty level	0.221	0.170	4.03	High	4.60	High
Practices	0.324*	0.033	4.29	High	4.65	High
Specific operations	0.326*	0.047	3.77	High	4.10	High
pair programming	0.555**	0.001	3.74	High	4.25	High
Cooperation using discussion forums	0.455**	0.006	4.06	High	4.35	High
Motivation students through the block-based collaborative technology	1.000	-	4.16	High	4.45	High
Valid N (listwise)			31		19	
	**.p<0.01 level.					
	*.p<0.05 level.					

In order to evaluate the correlation between two different variables, which take value in the set (1, 2, 3, 4, 5), are nominal and ordinal. We use Kendall's coefficient tau (τ). This statistic measures the correlation between two values, and it gives a value range between -1 and 1, where 0 indicates no correlation, -1 indicates a perfect negative correlation, and 1 indicates a perfect positive correlation. Kendall's tau is similar to other commonly used correlation coefficients such as Pearson's and Spearman's. The benefits of using Kendall's tau include more accurate statistical properties and a clearer interpretation of the probabilities of observing consistent (concordant) and inconsistent (discordant) pairs. Additionally, Kendall's tau and Spearman's rank correlation coefficient often lead to the same conclusions in most scenarios [38].

The statistical analysis demonstrates that there is a significant and positive relationship between easy debugging, simplicity, pair programming, and cooperation with discussion forums, and the motivation of students to improve their learning for sig (0.000) which $p < 0.01$. Additionally, there is a positive correlation between practices and object representation, and student motivation to enhance their learning for sig (0.000) which $p < 0.05$. The correlation coefficient is the highest for easy debugging (0.594), and it is the least for variable specific operations (0.326). However, there is a weak correlation relationship between difficulty level and student motivation to enhance their learning of the object-oriented course ($tau = 0.221$, $p = 0.170$). We believe that the difficulty level factor is highly beneficial for students, as evidenced by the high average obtained (4.03). However, the lack of a relationship with motivation may be attributed to negative experiences with previous tasks for some students, causing them to be hesitant to engage with this factor. We suggest that teachers or educational environments provide tasks that are well-suited to the abilities of their students [39].

In conclusion, most of the programming learning factors, which were based on the findings of the prior study and interviews, were found to be favorably accepted from both perspectives. However, there are still some things that can be done to enhance students' learning of object-oriented programming. Firstly, students should be encouraged to work together and practice as much as they can. Secondly, they should be encouraged to increase the level of competition between themselves. Lastly, the educational environments must be made enjoyable and fun for students.

6. CONCLUSION

This study provides a comprehensive overview of the key factors that facilitate the learning of object-oriented programming. The study employed two evaluation methods: interviews and surveys. The resulting factors that can aid students in learning object-oriented programming are: i) easy debugging, ii) object representation, iii) simplicity, iv) practice, v) specific operations, vi) collaborative programming, vii) difficulty level, and viii) possibility for cooperation. A questionnaire was then administered to assess the effectiveness of these factors in facilitating object-oriented programming learning.

ACKNOWLEDGEMENTS

This study is supported by the Universiti Kebangsaan Malaysia under research grant scheme of GUP-2020-090.

REFERENCES

[1] J. Szydlowska, F. Miernik, M. S. Ignasiak, and J. Swacha, "Python programming topics that pose a challenge for students," in *Third International Computer Programming Education Conference (ICPEC 2022)*, 2022.

[2] L. Bashiru and A. A. Joseph, "Learning difficulties of object oriented programming (OOP) in University of Ilorin-Nigeria: students perspectives," in *Proceedings of the Eighth TheIIER-Science Plus International Conference, Dubai, United Arab Emirates*, 2015, pp. 1–45.

[3] N. F. M. Sani, A. M. Zin, and S. Idris, "Analysis and design of object-oriented program understanding system," *International Journal of Computer Science and Network Security*, vol. 9, no. 1, pp. 125–134, 2009.

[4] Q. Batiha, N. Sahari, N. Aini, and N. Mohd, "Adoption of visual programming environments in programming learning," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 12, no. 5, Sep. 2022, doi: 10.18517/ijaseit.12.5.15500.

[5] M. Kölling, N. C. C. Brown, H. Hamza, and D. McCall, "Stride in BlueJ-computing for all in an educational IDE," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, Feb. 2019, pp. 63–69, doi: 10.1145/3287324.3287462.

[6] M. Kölling, "Blue, BlueJ, Greenfoot," in *Innovative Methods, User-Friendly Tools, Coding, and Design Approaches in People-Oriented Programming*, IGI Global, pp. 42–87, doi: 10.4018/978-1-5225-5969-6.ch002.

[7] A. Ferrari, G. Lombardo, M. Mordonini, A. Poggi, and M. Tomaiuolo, "OOPP: Tame the design of simple object-oriented applications with graphical blocks," in *Smart Objects and Technologies for Social Good*, Springer International Publishing, 2018, pp. 279–288, doi: 10.1007/978-3-319-76111-4_28.

[8] J.-M. Su and F.-Y. Hsu, "Building a visualized learning tool to facilitate the concept learning of object-oriented programming," in *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, Jul. 2017, pp. 516–520, doi: 10.1109/IIAI-AAI.2017.180.

[9] A. C. Bart, J. Tibau, E. Tilevich, C. A. Shaffer, and D. Kafura, "BlockPy: An open access data-science environment for introductory programmers," *Computer*, vol. 50, no. 5, pp. 18–26, 2017, doi: 10.1109/MC.2017.132.

[10] M. A. Bakar, M. Mukhtar, and F. Khalid, "The effect of turtle graphics approach on students' motivation to learn programming: a case study in a Malaysian University," *International Journal of Information and Education Technology*, vol. 10, no. 4, pp. 290–297, 2020, doi: 10.18178/ijiet.2020.10.4.1378.

[11] A. V Robins, "12 novice programmers and introductory programming," *The Cambridge handbook of computing education research*, 2019.

[12] Y. Soepriyanto and D. Kuswandi, "Gamification activities for learning visual object-oriented programming," in *2021 7th International Conference on Education and Technology (ICET)*, Sep. 2021, pp. 209–213, doi: 10.1109/ICET53279.2021.9575076.

[13] K. J. Mackin, "Turtle graphics for early Java programming education," *Artificial Life and Robotics*, vol. 24, no. 3, pp. 345–351, Sep. 2019, doi: 10.1007/s10015-019-00528-y.

[14] M. S. Naveed and M. Sarim, "Two-phase CS0 for introductory programming: CS0 for CS1," *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences*, vol. 59, no. 1, pp. 59–70, 2022.

[15] B. T. Fasy, S. A. Hancock, B. Z. Komlos, B. Kristiansen, S. Micka, and A. S. Theobold, "Bring the page to life: engaging rural students in computer science using alice," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, Jun. 2020, pp. 110–116, doi: 10.1145/3341525.3387367.

[16] A. Al-Jarrah and E. Pontelli, "'AliCe-ViLlagE' Alice as a collaborative virtual learning environment," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, pp. 1–9, doi: 10.1109/FIE.2014.7044089.

[17] A. A. Allinjawi, H. A. Al-Nuaim, and P. Krause, "Evaluating the effectiveness of a 3d visualization environment while learning object oriented programming," *Journal of Information Technology and Application in Education*, vol. 3, no. 2, 2014, doi: 10.14355/jitae.2014.0302.01.

[18] B. Selwyn-Smith, C. Anslow, M. Homer, and J. R. Wallace, "Co-located collaborative block-based programming," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2019, pp. 107–116, doi: 10.1109/VLHCC.2019.8818895.

[19] A. Feng, M. Gardner, and W. Feng, "Parallel programming with pictures is a Snap!," *Journal of Parallel and Distributed Computing*, vol. 105, pp. 150–162, Jul. 2017, doi: 10.1016/j.jpdc.2017.01.018.

[20] B. Broll and A. Ledeczi, "Distributed programming with NetsBlox is a Snap! (abstract only)," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Mar. 2017, doi: 10.1145/3017680.3022379.

[21] K. K. Agarwal, A. Agarwal, and L. Fife, "Python and visual logic-a good combination© for CS0," *Journal of Computing Sciences in Colleges*, vol. 27, no. 4, pp. 22–27, 2012.

[22] N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen, and R. Lysecky, "Python versus C++ an analysis of student struggle on small coding exercises in introductory programming courses," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 86–91.

[23] C. S. Miller, A. Settle, and J. Lalor, "Learning object-oriented programming in python," in *Proceedings of the 16th Annual Conference on Information Technology Education*, Sep. 2015, pp. 59–64, doi: 10.1145/2808006.2808017.

[24] C. S. Miller and A. Settle, "Some trouble with transparency," in *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 2016, pp. 133–141, doi: 10.1145/2960310.2960327.

[25] T. S. McNerney, "Tangible programming bricks: An approach to making programming accessible to everyone," Massachusetts Institute of Technology, 1999.

[26] T. McNerney, "From turtles to tangible programming bricks: explorations in physical language design," *Personal and Ubiquitous Computing*, vol. 8, no. 5, Sep. 2004, doi: 10.1007/s00779-004-0295-6.

[27] S. A. Papert, *Mindstorms: Children, computers, and powerful ideas*. Basic books, 2020.

[28] F. Masterson, "Evaluating logo:," *Computers in the Schools*, vol. 2, no. 2–3, pp. 179–195, Jul. 1985, doi: 10.1300/J025v02n02_20.

[29] F. L. Khaleel, N. S. Ashaari, T. S. M. Tengku Wook, and A. Ismail, "Programming learning requirements based on multi perspectives," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 3, pp. 1299–1307, Jun. 2017, doi: 10.11591/ijece.v7i3.pp1299-1307.

[30] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. Harper and Row New York, 1990.

[31] D. J. DeRose and B. Laurel, "Computers as theatre," *TDR (1988-)*, vol. 37, no. 4, p. 175, 1993, doi: 10.2307/1146303.

[32] M. S. Horn and R. J. K. Jacob, "Designing tangible programming languages for classroom use," in *Proceedings of the 1st international conference on Tangible and embedded interaction*, Feb. 2007, pp. 159–162, doi: 10.1145/1226969.1227003.

[33] R. Latih, M. A. Bakar, N. Jailani, N. M. Ali, S. M. Salleh, and A. M. Zin, "PC 2 to support instant feedback and good programming practice," in *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, Nov. 2017, pp. 1–5, doi: 10.1109/ICEEI.2017.8312410.
[34] C. Jost and B. Le Pévédic, "Designing evaluations: researchers' insights interview of five experts," in *Springer Series on Bio- and Neurosystems*, Springer International Publishing, 2020, pp. 287–330, doi: 10.1007/978-3-030-42307-0_12.
[35] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, "Learning difficulties in programming courses: Undergraduates' perspective and perception," in *2009 International Conference on Computer Technology and Development*, 2009, pp. 42–46, doi: 10.1109/ICCTD.2009.188.
[36] D.-Y. Kwon, H.-S. Kim, J.-K. Shim, and W.-G. Lee, "Algorithmic bricks: A tangible robot programming tool for elementary school students," *IEEE Transactions on Education*, vol. 55, no. 4, pp. 474–479, Nov. 2012, doi: 10.1109/TE.2012.2190071.
[37] J. Jacoby and M. S. Matell, "Three-point likert scales are good enough," *Journal of Marketing Research*, vol. 8, no. 4, pp. 495–500, Nov. 1971, doi: 10.1177/002224377100800414.
[38] J. Haigh and W. J. Conover, "Practical nonparametric statistics," *Journal of the Royal Statistical Society. Series A (General)*, vol. 144, no. 3, 1981, doi: 10.2307/2981807.
[39] R. H. Shroff and D. R. Vogel, "Assessing the factors deemed to support individual student intrinsic motivation in technology supported online and face-to-face discussions," *Journal of Information Technology Education: Research*, vol. 8, pp. 59–85, 2009, doi: 10.28945/160.

BIOGRAPHIES OF AUTHORS

Qais Ali Batiha received the B.Sc. degree in Computer Science from Jordan University of Science and Technology in 2010 and the M.S. in Information Technology from UUM in 2014. Currently, he is a Ph.D. student at the Faculty of Information Science and Technology at Universiti Kebangsaan Malaysia. He can be contacted at email qais_bateeha@yahoo.com, or P93243@siswa.ukm.edu.my.

Nazatul Aini Abd Majid she did her Bachelor of Computer Science with Honours and MSc (Computer Science) at Universiti Kebangsaan Malaysia and received her PhD from The University of Auckland, New Zealand in 2011. Currently she is a senior lecturer and researcher at the Universiti Kebangsaan Malaysia. Her research center is center for artificial intelligence technology. Her research interest includes augmented reality, educational robotic, multivariable statistical process monitoring, industrial computing, high performance computing, cloud computing. She can be contacted at email nazatulaini@ukm.edu.my.

Noraidah Sahari currently she is a senior lecturer and researcher at the Universiti Kebangsaan Malaysia. Her research interest includes Multimedia application e-learning technology interaction design and usability. She can be contacted at email nsa@ukm.edu.my.

Noorazean Mohd Ali currently she is a senior lecturer and researcher at the Universiti Kebangsaan Malaysia. Her research interest includes aspect-oriented programming and design, programming education, object-oriented design and development, computational thinking. She can be contacted at email aliazean@ukm.edu.my.