Applied Mathematics and Computational Intelligence
Volume 11, No.1, Dec 2022 [231-242]

Factors Affecting the Educator's Ability on Learners' Attraction in Learning the Programming Subject Specifically to the Engineering Students at UiTM Pulau Pinang

Jamal Othman[1]*, Siti Balqis Mahlan[2], Rozita Kadar[3], Maisurah Shamsuddin[4] and Naemah Abdul Wahab[5]

[1,2,3,4,5]Department of Computer and Mathematical Sciences
Universiti Teknologi MARA Cawangan Pulau Pinang, Malaysia

*Corresponding author: jamalothman@uitm.edu.my

Received: 28 April 2022; Accepted: 13 September 2022; Available online (In press): 30 September 2022

ABSTRACT

Teaching the subject of computer programming, especially to engineering students at higher education institutions, requires a comprehensive teaching approach, flexible pedagogy and hybrid teaching creativity to cultivate learners' attraction and realise a realistic learning atmosphere. Research demonstrated that students are not attentive to learning the programming subject owing to several factors such as teaching materials and pedagogy, class size and programming difficulties. Hence, this paper concentrates on the factors related to educators contributing to the students' attraction to learning the programming subject. This study was conducted at Universiti Teknologi MARA, Pulau Pinang Branch with a total of 241 students from the engineering schools responding to the online survey. The data collected were analysed using descriptive statistics and factor analysis. The reliability test or Cronbach's Alpha values was performed before analysing the questionnaire. The IBM SPSS was used in evaluating the data collected from the questionnaire and through factor analysis, it was found that the educators' roles affected the students' attraction and understanding ability in learning the programming subject. The results obtained from factor analysis indicated that the personality traits of programming lecturers and instructional material (additional references) are the major contributors to the attraction of students toward programming subjects. This finding can help educators to improvise and innovate new teaching approaches to make the computer programming class fascinating and enhance the students' learning.

Keywords: educators, teaching, computer programming, engineering student, programming problem.

1 INTRODUCTION

The subject of computer programming is one of the core papers offered to all engineering students at any higher education institution in Malaysia. Failure to complete this subject will lead the students to be unable to graduate and unqualified to obtain their degree or certificate from the university. The role of educators is important in making the computer programming class effective and exciting. Researchers reported that the introductory programming subjects offered in the computer science curriculum at tertiary education institutions are facing a universal problem. It has a high dropout and failure rate among students [1], [2].

According to Cheah [3], despite the variety of programming tools available in teaching the programming subject, students' performance continues to decline. One of the critical reasons is due to the lack of students' ability in problem-solving. Furthermore, critical thinking skills are low among students at the tertiary education level, which contributes to the decline in programming subject performance. Based on past research conducted by Ismail et al. [4], a higher level of knowledge of 'when' and 'why' from the metacognitive skills is needed during the first stage of programming education. Besides, static teaching materials such as printed book references with unattractive presentations or textual explanations without infographics are ineffective teaching materials for learning the dynamic nature of computer programming subject [5].

Hence, this paper concentrates on the educator factors influencing the performance in the programming subject among engineering students at Universiti Teknologi MARA (UiTM), Penang branch. This research was intentionally carried out to assist and guide computer science educators to improvise their teaching methodology for computer programming, as well as improve students' interest and performance.

1.1 Roles of Educators in Teaching Programming

Programming is a complex subject that requires continuous effort, special approach and multi-layer skills [6]. The roles of educators are extremely important for making the programming class interesting and fun. The teaching methods using conventional static materials such as the textbook, marker pen and slide do not raise learners' understanding [5]. Moreover, senior educators still sustain the same static teaching approach without improvising their teaching style or methodology aligned with the current demands of education revolution, which will discourage the students' interest [7], [8]. Conventional approaches are appropriate if the class is a combination of several groups of learners and handled as a large crowd for lectures, which involves more than 50 to 100 students. Teaching the programming subject requires physical interaction between the educator and learners besides the creation of dynamic communication and understanding of programming concepts. The educators can give immediate feedback to the learners when the class is divided into small groups; this approach looks ideal as detailed explanations can be provided whenever needed by the learners [9]. Interactivity in the class with the elements of spatial and visualisation are much more effective than conventional static programming materials such as hardcopy and softcopy notes. Thus, using the conventional approach will degrade the learning curve among learners, dropping the learners' confidence in the programming subject.

Some educators are concerned about the syntax of the programming language rather than understanding the problem-solving methods. This is because the curriculum was designed to focus on the popularity of the programming language with the current demands of the industrial revolution (IR) 4.0. Gomes [10] elaborated that the suitability of pedagogy for teaching the programming subject has been put aside by the curriculum designers while the popularity of programming language is prioritised. Selecting inappropriate programming language and teaching pedagogy will affect the effectiveness of learning programming and negatively impact learners' understanding of the subject [11]. As a result, the learners will be unable to apply the programming concepts in real-life or problem-solving. This is agreed by Byrne and Lyons [12] stating that the learners who face difficulties in mastering the concepts of numbers theory, calculus, geometric and trigonometric will fail to transform the abstract or problem statements into mathematical formulas.

The instructors should be competent in the demands of high-level abstraction and analytical thinking to produce comprehensive solutions for any problem statements [13]. Hence, selecting a programming language should be less complex, easy to remember and improve the learning curve of learners during self-explanatory or self-study without formal guidance by the educators. The selection based on the popularity of the programming language with current industrial demands is not an excellent choice since the curriculum was designed according to education policy standards and not for professional purposes [10]. The learners should not only concentrate on learning the programming syntax. Understanding the programming semantics concept is fundamental to any programming language.

The research conducted by Ismail et al. [7] identified the main problem in teaching computer programming, which is the ineffective use of presentation techniques for problem-solving. Most educators still sustain the pseudocode and flowchart to explain to students on problem-solving steps. These tools are only applicable to structured programming. Nevertheless, the conventional tools are inappropriate for the object-oriented programming language, unless the educators know how to apply the Unified Modelling Language (UML) class-diagram tools for an object-oriented approach. Approaches that provide more visualisation in explanation are needed to allow the students to have a mental representation of a given problem [13].

Table 1: Summary of Problems Faced by the Educators in Teaching the Programming Subject

Problem	Author(s)	Descriptions
Using the conventional static materials	[5]	• The educators use the textbook, marker pen and slide for teaching purposes. • The teaching delivery is not interesting, creative and effective enough to create students' interest and awareness.
The class should be divided into small groups	[9]	• Detailed explanations can be done effectively for students. • Creates interactivity in the class that promotes the elements of spatial and visualisation in the learners.
Educators are concerned with the programming syntax instead of the problem-solving	[10]	• For fulfilling the current demands of the industrial revolution (IR) 4.0 • Selecting popular programming languages based on the current market popularity.
Selecting the wrong pedagogy and programming approaches	[11][12][7] [8][4]	• Selecting inappropriate programming language and teaching pedagogy, such as using the structured approach including pseudocode and flowchart for Object-Oriented Programming (OOP). Supposedly, educators should apply the Unified Modelling Language (UML) for OOP.
The programming language selected is difficult to teach	[10][13]	• Instructors should be competent in the demands of high-level abstraction and analytical thinking. • Programming language should be less complex with easy-to-remember syntaxes.

	• Understanding the programming semantics is fundamental for any programming language.

2 METHODOLOGY

This study involved 241 students; 149 of them were diploma students from the Faculty of Mechanical Engineering (FKM) while the remaining were degree students from the Faculty of Civil Engineering (FKA) who took the programming subject at the UiTM Pulau Pinang branch. Table 2 below displays the number of students who took programming language by semester. For the diploma level, students took this subject in semester 2, while for the degree level, students took programming language in semester 2 or semester 4. The remaining were students who repeat the subject.

Table 2. Number of Students by Semester

	Programming Code	
Semester	**CSC128 (Diploma) Mechanical Engineering**	**CSC425 (Degree) Civil Engineering**
2	142	12
3	1	3
4	0	76
5	1	1
6	4	0
8	1	0
TOTAL	**149**	**92**

This study's questionnaire was divided into two sections. The first section concerned the course or subject information that was taken, whereas the second section involved the educator's factors associated with the students. All questions contained 14 items and were divided into three sections as indicated in the table below (Table 3 and Table 4).

Table 3: Construct Questions in Course or Subject Information

Construct	**Options**
Program Code	1. *Mechanical Engineering* 2. *Civil Engineering*
Study Level	1. *Diploma* 2. *Degree*
Semester	*1/2/3/4/5/6/7/8*
Programming Code Taken	1. *CSC128* 2. *CSC425*

Status Taken	1. *First Timer* 2. *Not First Timer*

Table 4. Construct Questions in Educator-Related Factors

Construct	**Statements**
A. Personality Traits of My Programming Lecturer …	1. *Has a good relationship with the students.*
	2. *Displays smartness, confidence and firmness in making decisions.*
	3. *Enforces proper discipline and is strict in following prescribed rules.*
	4. *Has an interesting personality with a good sense of humour.*
	5. *Is open to suggestions and opinions and is worthy of praise.*
B. Teaching Skills of My Programming Lecturer…	1. *Explains the objective of the lesson clearly at the start of each class.*
	2. *Has mastery of the subject matter.*
	3. *Is organised in presenting subject matters by systematically following the course or subject outline.*
	4. *Is updated with present trends and relevant to the subject matter.*
	5. *Uses various strategies, teaching aids/devices and techniques in presenting the lessons.*
C. Instructional Materials of My Programming Lecturer…	1. *'Chalk and blackboard' in explaining the lesson.*
	2. *Workbooks/textbook.*
	3. *Visual aids (e.g., PowerPoint).*
	4. *Articles/material/notes/hand-outs for additional references.*

A reliability Test or Cronbach's Alpha was first performed before analysing the questionnaire. Reliability describes how reliable and consistent is a research instrument's measurement of a variable. The better the instrument's reliability, the fewer errors it generates [14]. Cronbach's Alpha values are based on [15].

Cronbach's Alpha was used in this analysis to measure the internal consistency of the items tested. According to Table 5, Cronbach's alpha value for all 14 questionnaires tested was 0.889. This value was greater than 0.8, which is considered reliable.

Table 5. Reliability Test

Cronbach's Alpha	**N of items**
.889	14

Five-point Likert scale was employed in these questionnaires. The educator's related factors (Construct A and B) used the range from 5-strongly agree to 1-strongly disagree. While Construct C used the range from 5-always, 4-often, 3-sometimes, 2-rarely, and 1-never; values greater than 3 are positive and values less than 3 are negative statements. Figure 1 shows the students' responses to Construct A: Personality Traits of My Programming Lecturer. It was found that the number of students who strongly agreed with the statement was quite high, which was represented by the values of 66.8%, 66.4%, 54.4%, 57.7% and 62.7%, respectively.

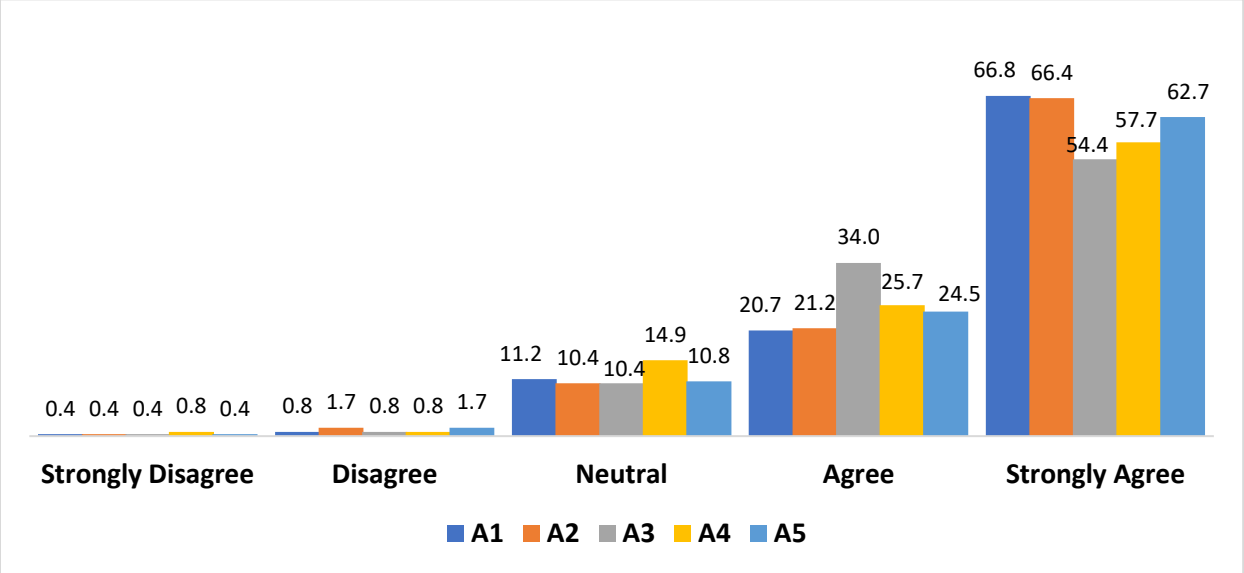

Figure 1. Percentage of Students' Responses for Construct A (Personality Traits of My Programming Lecturer)

Figure 2 illustrates the students' responses to Construct B: Teaching Skills of My Programming Lecturer. It was also found that the response of the students was also quite high for those who strongly agreed with the statement shown by the values of 66.8%, 66.4%, 54.4%, 57.7% and 62.7%, respectively. All constructs B1, B2, B3, B4 and B5 were found to only contribute fewer than 3% of those who responded disagreed and strongly disagreed with the statement.

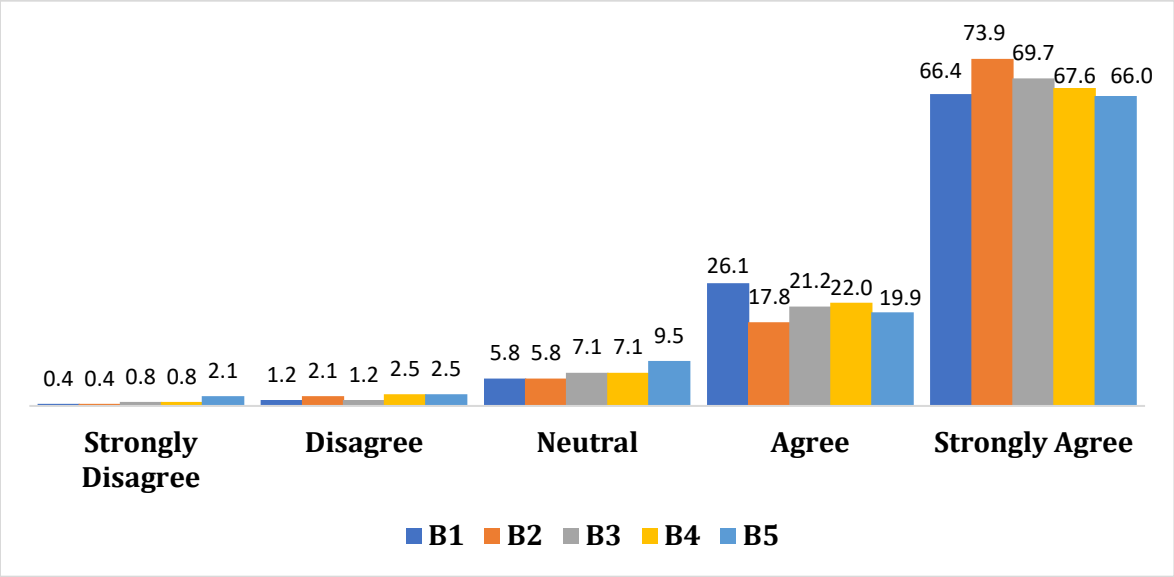

Figure 2. Percentage of Students' Responses for Construct B (Teaching Skills of My Programming Lecturer)

It can be seen in Figure 3 below that for the "always" scale, 52.3% and 47.7% of students prefer visual aids (e.g. PowerPoint) and Articles/Materials/Notes/Hand-outs for additional references. Furthermore, there was a relatively similar response in each Likert scale for the selection of the Chalk and Whiteboard method where the percentage was around 20% to 27% except for the "rarely" scale, which has given a percentage of 9.1%.

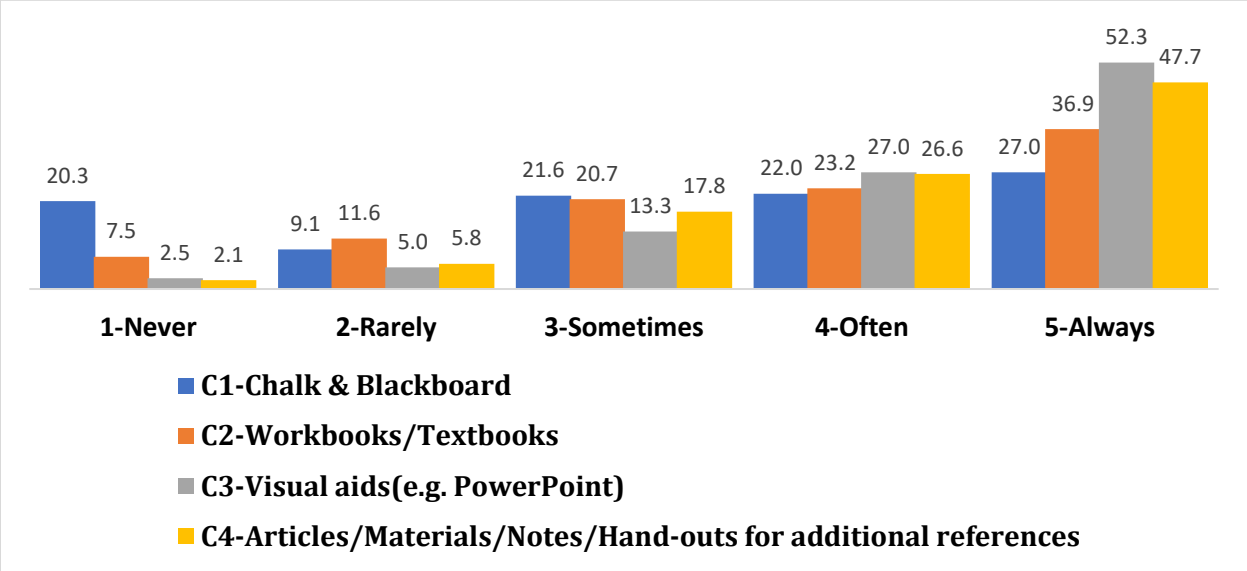

Figure 3. Percentage of Students' Responses for Construct C (Instructional Materials of My Programming Lecturer)

The data collected were analysed using Factor Analysis (FA) to find the major factor contributing to the attraction of students towards the programming subject. FA is only acceptable for use when the KMO value is greater than 0.5 [16]. Then, to ensure that the present example contains patterned relationships, Barlett's Test of sphericity was performed. To proceed with the FA, the significant value must be less than 0.05. Next, the best number of factors will then be chosen at eigenvalues greater than one. When a factor has an eigenvalue less than 1.00, it contains fewer than one variable. Those with eigenvalues less than 1.00 are not considered to be stable [17]. After that, the factors were rotated using the varimax method to create the best factors. The main factor will then be determined at the end.

3 RESULTS AND DISCUSSION

In this paper, the data collected were analysed using FA to determine the suitability of all questions containing 14 items. To find out which factor contributes more to students' attraction to the programming subject, the data were further analysed using FA. Table 6 shows the results of KMO and Barlett's Test. The value of KMO obtained was 0.897, which was greater than 0.5. FA is only appropriate to be used when the KMO value is at least 0.5. KMO values of less than 0.5 should lead to more data collection or choosing variables to include. Low KMO indicates an insufficient sample size to continue the procedure. As a result, the argument may not be valid. The Barlett's Test obtained was 0.000, which was less than the predetermined value $\alpha = 0.05$. H_o: There is no correlation between variables and H_1: There is a correlation between variables; these are the hypotheses used to test Bartlett's Test. The obtained p-value was 0.000 (< 0.05), which was less than the predetermined value alpha = 0.05. As a result, H_o was rejected, since there was a relationship between variables. This result shows that the factor analysis could be done [16].

Table 6. KMO and Barlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.897
Bartlett's Test of Sphericity	Approx. Chi-Square	2254.756
	df	91
	Sig.	.000

According to Table 7, 14 items were identified before extraction using FA. After the extraction was done, 3 factors were considered as the factors based on eigenvalues greater than 1, where the eigenvalues were 7.125, 1.294 and 1.1111. It was found that the total cumulative variances for these 3 factors were 68.071%. A total of 50.892% was explained by the first factor, 9.246% was explained by the second factor and 7.933% was explained by the third factor after the stage of extraction.

At the stage of rotation, the eigenvalues must be higher than 1 similar to the extraction method. The variance for the first factor was reduced from 50.892% to 29.236%, which was by 21.656%. However, the variances of the other two factors increased by about 47.719% and 60.138%. These confirmed that the first factor was more correlated to the dependent variable compared to other factors.

Table 7. Total Variance Explained

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings		
	Total	% of Var	Cum %	Total	% of Var	Cum %	Total	% of Var	Cum %
1	7.125	50.892	50.892	7.125	50.892	50.892	4.093	29.236	29.236
2	1.294	9.246	60.138	1.294	9.246	60.138	3.882	27.726	56.962
3	1.111	7.933	68.071	1.111	7.933	68.071	1.555	11.108	68.071
4	.873	6.233	74.304						
5	.741	5.296	79.600						
6	.614	4.384	83.984						
7	.452	3.229	87.213						
8	.376	2.686	89.899						
9	.369	2.638	92.537						
10	.339	2.424	94.960						
11	.254	1.816	96.776						
12	.206	1.470	98.246						
13	.142	1.012	99.258						
14	.104	.742	100.000						

According to Table 8, the factors influencing students' interest in programming subject were divided into 3 based on priority. Items A1, A2, A3, A4, A5 and C4 were the major factors to attract students in learning the programming subject. The second highest factor consisted of items B1, B2, B3, B4, B5 and C3. The last contributor comprised the items of C1 and C2.

Table 8. Rotated Component Matrix

Component	**1**	**2**	**3**
A1	0.724		
A2	0.845		
A3	0.786		
A4	0.821		
A5	0.821		
C4	0.396		
B1		0.736	
B2		0.821	
B3		0.829	
B4		0.772	
B5		0.711	
C3		0.508	
C1			0.822
C2			0.790

Based on Figure 4, all of the components that have been divided into three parts of factors and ranked in order of priority can represent factors that influence students' perceptions of programming subject. From Figure 4, factors A1, A2, A3, A4, A5 and C4 were the most influencing factors for learning the programming subject.

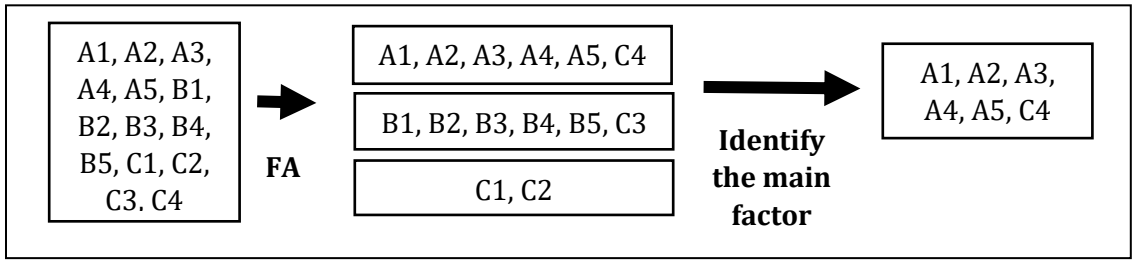

Figure 4. Summarisation of Factor Analysis (FA)

The reliability analysis is performed after the main factor has been determined. The Cronbach's Alpha value demonstrates how accurate the main factors are measured. Based on the results, it was found that Cronbach's Alpha value for the main factor, which is the combination between constructs A1, A2, A3, A4, A5 and C4, was 0.886. The Cronbach's Alpha value is acceptable considering that the value was greater than 0.8; the reliability level is excellent if the value is around 0.8 and 1.00. Cronbach's Alpha values are based on [15].

Similarly, previous studies [18] that used the regression analysis method investigated the factors influencing the learner's attraction to learn the programming subject and discovered that student-related factors (study habits) and programming ability-related factors contributed to programming learning. Whereas in this study, different methods were proposed to classify the factors, with factor analysis being the method used, and this study only focused on determining the factors based on educator-related factors.

4 CONCLUSION

The results of this study found that the major factors that most influence learning programming subjects are A1 (good relationship with the students), A2 (smartness, confidence and firmness in making decisions), A3 (proper discipline and strict), A4 (interesting personality), A5 (open to suggestions) and C4 (always use Articles/materials/notes/hand-outs for additional references). All of these factors affect the students' attraction and understanding ability in learning the programming subject. In conclusion, lecturers should give more priority to improving their personality traits. In addition, lecturers also need to emphasise the learning materials used, that is, by regularly using additional reference materials such as articles, other additional materials other than textbooks, additional notes and exercises. Hence, lecturers are suggested to follow any strengthening course that can enhance better personality traits to attract students to learn computer programming. Lecturers are also advised to create a module that contains appropriate additional references to learn computer programming more effectively.

Moreover, educators should dynamically improvise their teaching materials and teaching delivery to make the class more attractive, effective and able to enrich the programming knowledge. In addition, the skills of personal traits of the instructors are also among the contributing factors that enable enhancing teaching effectiveness. To measure the student's understanding or ability to remember the lectures, the educators should conduct simple online pop quizzes with the students 5 minutes before the class ends. Through this simple assessment, educators can measure their teaching effectiveness. Educators can focus on the root causes such as lack of problem-solving skills and critical thinking ability, which contribute to the student's performance in the programming subject. The educators should imply and execute dynamic problem-solving related to the student's daily life or environment and put themselves in their students' shoes and slowly work together with them in solving the real problems, step by step until everybody can continue solving the problem individually without any help from the educators.

REFERENCES

[1] A. Luxton-Reilly, "Learning to program is easy," Paper presented at the Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, 2016.

[2] A. Robin, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Computer Science Education,* vol. 13, no. 2, pp. 137-172, 2003.

[3] C. S. Cheah, "Factors contributing to the difficulties in teaching and learning of computer programming: A literature review," *Contemp. Educ. Technol.*, vol. 12, no. 2, pp. 1–14, 2020, doi: 10.30935/cedtech/8247.

[4] M. N. Ismail, N. A. Ngah, and I. N. Umar, "The effects of mind mapping with cooperative learning on programming performance, problem solving skill and metacognitive knowledge among computer science students," *Journal of Educational Computing Research,* vol. 42, no. 1, pp. 35-61, 2010.

[5] J. Bennedsen and M. E. Caspersen, "Revealing the programming process," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, 2005, pp. 186–190.

[6] H. C. Jiau, J.C. Chen, and K.F. Ssu, "Enhancing self-motivation in learning programming using game-based simulation and metrics.," *IEEE Transactions on Education,* vol. 52, no. 4, pp. 555-562, 2009.

[7] M. N. Ismail, N. A. Ngah, and I. N. Umar, "Instructional strategy in the teaching of computer programming: a need assessment analyses," *TOJET Turkish Online J. Educ. Technol.*, vol. 9, no. 2, pp. 125-131, 2010.

[8] Y. Boose and M. A. Gerosa, "Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage," *ACM SIGSOFT Software Engineering Notes*, 41(6), pp. 1-6, 2017.

[9] X. Zhang, C. Zhang, T. F. Stafford, and P. Zhang, "Teaching introductory programming to IS students: The impact of teaching approaches on learning performance," *J. Inf. Syst. Educ.*, vol. 24, no. 2, pp. 147–155, 2013.

[10] A. Gomes and A. J. Mendes, "Learning to program-difficulties and solutions," in *International Conference on Engineering Education–ICEE*, 2007, vol. 7.

[11] N. C. C. Brown and G. Wilson, "Ten quick tips for teaching programming," *PLoS Comput. Biol.*, vol. 14, no. 4, p. e1006023, 2018.

[12] P. Byrne and G. Lyons, "The effect of student attributes on success in programming," in *Proceedings of the 6th annual conference on Innovation and technology in computer science education*, 2001, pp. 49–52.

[13] A. V. Robins, "Novice Programmers and Introductory Programming," *Cambridge Handb. Comput. Educ. Res.*, pp. 327, 2019.

[14] R. Kumar, Research methodology: A step-by-step guide for beginners. Sage, 2018.

[15] N. Choi, D. R. Fuqua, and B. W. Griffin, "Exploratory analysis of the structure of scores from the multidimensional scales of perceived self-efficacy," *Educ. Psychol. Meas.*, vol. 61, no. 3, pp. 475–489, 2001.

[16] F. G. Kaiser, System Analysis by Digital Computer. Wiley, 1974.

[17] E. R. Girden, Evaluating Research Articles from Start to Finish. Sage Publications, 2001.

[18] R. Kadar, S. B. Mahlan, M. Shamsuddin, J. Othman, and N. Wahab, ''Analysis of factors contributing to the difficulties in learning computer programming among non-computer science students,'' in *IEEE 12th Symposium on Computer Applications & Industrial Electronics*, 2022, pp. 89-94.