CONTEMPORARY EDUCATIONAL TECHNOLOGY

ISSN: 1309-517X (Online) 2020, 12(2), ep272, https://doi.org/10.30935/cedtech/8247

OPEN ACCESS

Research Article

Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review

Chin Soon Cheah
School of Educational Studies, Universiti Sains Malaysia, 11800, USM Penang, Malaysia
ORCID: 0000-0002-6012-5660

Received: 9 Mar 2020 Accepted: 29 Apr 2020

Abstract

This paper reviews the literature relating to the factors that contribute to the difficulties in learning of computer programming. Programming has been a difficult subject to learn and master even at the early stage of education. It has been a global problem and continues to worsen at the local level. Although, there are many education tools available to complement the teaching and learning of computer programming. The problem persists until present day. Even at the initial stage of computer introduction courses, there were high failure rates and high drop out. One of the rationalism behind this scenario is from the students' lack of problem solving abilities. However, the problem does not only lie within the students' level of efficacy. It involves the effective use of teaching and learning material as well. Therefore, the focus of this review is on the factors concerning the students learning method and the effectiveness of the teaching material. Based on plethora of literature review, it is vital that these two aspects need to be consider simultaneously to overcome the difficulties of learning computer programming. A much more comprehensive, effective and universal teaching and learning tools need to be design to address the severity of this problem.

Keywords: computer programming, difficulties in programming, phases of programming, teaching materials

INTRODUCTION

Globally, programming has been a universal problem in the computer science curriculum. It has high dropout and failure rates even do at the initial stage of computer introduction courses (Luxton-Reilly, 2016; Robins, Rountree, & Rountree, 2003). Many tools are available that can assist the teaching and learning of programming but, the problems remain unresolved. In terms of students' aspect, many of them lack problem-solving abilities (Bosse & Gerosa, 2017; Gomes & Mendes, 2007; Robins, 2019; Savage & Piwek, 2019). Programming is a complex subject that requires continuous effort, special approach and multi-layer skill. The process of obtaining these skills is a tedious trial-and-error process and persistency (Jiau, Chen, & Ssu, 2009). Series of abilities and persistency is needed to become a good programmer and knowing the syntax of programming is just the beginning of a challenging aspect of creating a good program. Students need to understand beyond the syntax and flow of computer programming in order to resolve complex real-world problem (Bosse & Gerosa, 2017).

Past research has shown that higher-level knowledge of 'when' and 'why' from the metacognitive skills is needed during the early stage of programming education (Ismail, Ngah, & Umar, 2010b). Besides that, static teaching material such as printed books were ineffective as teaching material for learning the dynamic nature of programming (Bennedsen & Caspersen, 2005; Jenkins, 2002). In terms of pedagogy, the programming

language chosen should be less complex (Gomes & Mendes, 2007). Moreover, students' negative perception influences students' attitudes toward learning computer programming. Thus, it impaired the intrinsic motivation of the students to continue learning for success (Ng & Bereiter, 1991; Qian & Lehman, 2017; Robins, 2019).

REVIEW METHOD

To obtain a comprehensive and detailed understanding of the factors that contribute to the difficulties in teaching and learning of computer programming, the literature searching went through rigorous steps to ensure quality and accurate information we found. Online databases such as EBSCOhost, Eric, ProQuest, SAGE, ScienceDirect, and Taylor & Francis we searched as they were a reliable source of channels in providing educational literature in terms of computing and education. Keywords such as "computer programming", "teaching and learning", "contributing factors", and "difficulties in computer programming" were used. In order to get a more precise and accurate information, combinations of the keywords and search operators such as AND, NOT, and OR were used to further narrow down the search results.

OVERVIEW

Based on the findings from past researchers, it has shown that problems do occur at the initial stage of learning computer programming. Students' were having difficulties in understanding abstract programming concepts such as control structures, and creating an algorithm to solve concrete problems (Gomes & Mendes, 2007; Luxton-Reilly, 2016; Robins, 2019). Moreover, lack of consolidation of knowledge and abilities in problem solving skills and logic reasoning further deteriorate the problems during the early stage of learning (Bosse & Gerosa, 2017; Savage & Piwek, 2019).

Hence, the focus should be on the higher-level knowledge such as programming strategy and conditional approach. The knowledge of 'when' and 'why' from the meta-cognitive skills should be taught and instill during the early stage of programming education. The metacognition consists of knowledge about when and how a certain approach can be used to solve a problem in the learning process (Ismail, Ngah, & Umar, 2010a). Several studies have shown that lack of meta-cognitive skills in computer programming courses are one of the contributing factors that made student unable to solve a programming problem (Chetty & Barlow-Jones, 2014; Chetty & van der Westhuizen, 2014; Holvikivi, 2010; Robins, 2019).

Traditional teaching method using statics material such as books, notes, and slide does not seem to be effective methods in teaching of computer programming (Bennedsen & Caspersen, 2005; Zhang, Zhang, Stafford, & Zhang, 2019). The reason behind the ineffectiveness of statics teaching material is that it does not personalized to certain group students, but rather to a large crowd of audience from different continents. Moreover, it does not provide live interaction and dynamic elements to explain programming concept. It would be ideal if an instructor is available to give immediate feedback, and detailed explanation whenever is needed by the students. However, this is impossible due to time limitations, manpower, and the large numbers of students available in a single class. Furthermore, the usage of static printed materials in teaching of programming is definitely a pitfall in explaining dynamic programming concepts (Zhang et al., 2019). Some students failed to understand the dynamic nature of programs whenever static materials were used to explain to them (Gomes & Mendes, 2007).

This is further supported by findings that show some students might prefer learning alone whereas some might prefer a dynamic learning environment (Jenkins, 2002). The dynamic learning environment mentioned can consists of group discussion and interaction between peers (Zhang et al., 2019). Therefore, the existing teaching method using static material does not support the students' different learning styles. Thus, it is important that the instructor ensures that the teaching methods provided are able to accommodate the different groups of students.

In terms of pedagogy, instructors are more focused on teaching the syntax of programming language rather than promoting problem solving method. Furthermore, the chosen programming language for teaching is

based on popularity rather than based on pedagogic suitability (Gomes & Mendes, 2007). By choosing an inappropriate programming language for educational purposes, it does not only impaired the effectiveness of learning computer programming, but it further increases the difficulties for the students to understand the subject (Brown & Wilson, 2018; Savage & Piwek, 2019).

In nature, programming demands high level of abstraction and analytical thinking in producing an effective solution. This required applying certain programming concepts and algorithms. Thus, the selection of programming languages based on industry popularity is not suitable as it was developed for professional use rather than supporting education purposes. The programming language that is chosen should be easy to remember, less complex and self-explanatory to ease the learning cycle during the introduction stage (Gomes & Mendes, 2007; Robins, 2019).

In addition, programming subjects required persistency, continues learning, and knowledge from other subjects from time to time. Students often ask for the solution or give up whenever they face an obstacle. Past experiments had been conducted and the result showed that the relationship between mathematical problem-solving competencies and the programming abilities do exist in the introduction of programming courses (Gomes, Carmo, Bigotte, & Mendes, 2006). This is further supported by Byrne and Lyons (2001) which concluded that students who face difficulties in understanding in calculus, number theory, geometric and trigonometric concepts leads to difficulties in transforming textual problem into mathematical formula due to weak foundation of understanding the basic concept .

Besides that, from the aspect of students' psychologically, students tend to have negative connotation relating to programming subjects. The reason behind the implication of negative thought relating to computer programming subjects is due to the passing of comments, opinion and suggestion from their senior peer who has taken the subjects before (Jenkins, 2002). By having a negative thought and wrong impression, students believe that programming will be difficult. Hence, its effects and diminish the intrinsic motivation and does motivate them to learn for success (Ng & Bereiter, 1991).

The severity of the negative connotation relating to computer programming does not only affect the intrinsic motivation, it influences the students' attitude on computer programming as well. An individual attitude is determined by one's behaviour and signifies the cognitive action which will determine the success or failure of an individual in achieving a certain task. Research done by Tüysüz (2010) shows that there is a positive correlation between attitude and achievement. It is further supported by the research done by Baser (2013) which has also shown a positive correlation between the students' attitude and their achievement in computer programming as well.

According to Tan, Ting, and Ling (2009), computer programming has been a tough subject for instructors to teach and students to learn. The difficulties in learning of computer programming consist of few factors. Initially, students find it difficult in designing a program to solve a certain task. This is due to the reason that the students were unable to divide different functionality into procedures. Furthermore, students struggle while learning the syntax of the programming languages which is difficult to remember and understand (Bosse & Gerosa, 2017; Qian & Lehman, 2017). Meanwhile, the process of eliminating bugs in the program that is time consuming, and tedious further diminished the motivation of the students in excelling to learn the programming languages.

According to the research done by Ismail et al. (2010b), lecturers have identified four main problems in computer programming education. The lack of skills in analysing problems and understanding the programming concepts is the first problem encountered in computer programming education. This is due to the reason that students lack of prerequisite skills such as discrete mathematics and logic programming course. The second contributing factor is caused by the ineffective use of presentation techniques for problem solving. Usages of conventional techniques such as pseudo code and flow charts are only suitable for teaching structured programming. When it comes to teaching object-oriented programming, the conventional techniques fail to provide adequate explanation and understanding to the students. Currently, most of the programming languages learn based on object-oriented programming. Approaches that provide

more visualisation in explanation are needed to enable the student to have a mental representation of the problem (Robins, 2019).

The third contributing problem arises from the ineffective use of teaching strategies for problem solving and coding techniques. Conventional teaching strategies are no longer effective in teaching object-oriented programming. Lecturers agree that different paradigms from the aspect of cognitive strategy should be used in teaching programming. Teaching material that supports spatial and visualisation abilities is needed to assist the student in understanding the process of control and data flow. Besides that, the lack of student active involvement and contribution during the practical session further deteriorate the problem. Hence, students failed to understand computer programming during the learning process (Ismail et al., 2010b).

The final contributor to the failure in learning of programming is that the students do not understand and did not master the programming syntax (Bosse & Gerosa, 2017; Qian & Lehman, 2017). The problem worsens when students are unable to code programming constructs as well. This can be further elaborate in terms of understanding the concept of programming, lack of knowledge in syntax, and unable to construct effective programming code (Robins, 2019).

DISCUSSION

Based on the plethora of literature mentioned above, there were large numbers of contributing factors in the difficulties in teaching and learning of computer programming. Generally, these contributing factors can be classified into four main factors and further analyse into sub-factors. The four main factors are phases of programming stage, problem solving skills, ineffective pedagogy, and personal traits and attitude.

Phases of Programming Stage

According to Dale and Weems (2005), programming can be divided into two phases and each phase required different types of knowledge and skills. For students to successfully master a programming language, an individual need to surpass each phase before proceeding to the next phase. It is vital that students grasp the knowledge and skills for each phase to avoid any errors at a later phase. These phases are the problem-solving phase, and the implementation phase.

Problem solving phase

Within the problem-solving phase, student needs to learn the concept of programming and data structure on how to analyse a problem. Students will be required to conduct analysis and specification steps to understand the problem. At this stage, understanding the type of variables, the usage of variables, and the numbers of variables needed to be used will be identified. This will be followed by generating a general solution which is called algorithm. Computer algorithms are English like statements that explained how and what should a computer programs execute in order to perform data processing. It contains step by step instructions and other functions such as calculation, data processing and automated decision making.

After developing an algorithm, the general solution needs to be verified and proceed to establish a solution called program. The verification process includes ensuring that the logic of the computer program flow is in correct sequence, the correct types of decision-making options are used, suitable and effective programming loops, and processing of each function does produce the correct output. All these steps are important as they will determine the final results. Therefore, cognitive ability such as reasoning and logical thinking is important at this stage to develop effective and robust algorithm.

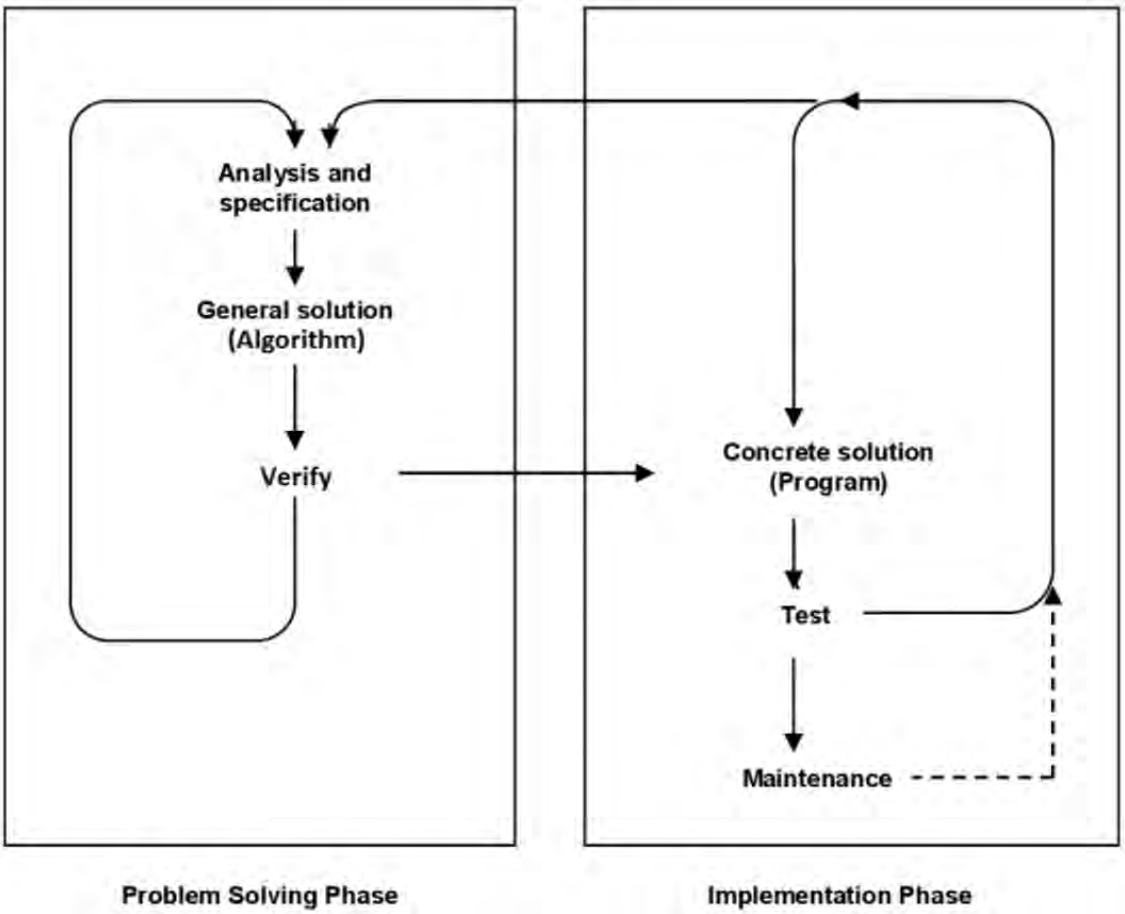

Figure 1. Programming Process (Dale & Weems, 2005)

Implementation phase

The creating of the solution stage marks the beginning of the implementation phase where the actual hands-on coding happens. In this stage, the actual programming code happens, and programmers will develop the code based on the information gathered during the problem solving phase. Once the program has been developed, extensive testing is conducted to verify the efficiencies of the program. Testing is done rigorously by using various types of input data and repetitive processing to ensure the system is stable. This is to ensure that the program behaves accordingly to the end objective of the program which is to solve a specific task without any errors. If the outcome of the testing does not meet the intended objective, an iterative of the problem solving phase will be taken place to further fine tune and correct the earlier stage of error (Dale & Weems, 2005).

Besides that, lack of understanding during the early stage of programming language constructs such as variables, arrays, recursion, and loops makes novice programmers unable to write effective functions and procedures (Luxton-Reilly, 2016; Savage & Piwek, 2019). Moreover, this has contributed to the reason which enable them to conduct effective testing (Soloway & Spohrer, 2013). Hence, programming required extensive persistency and tedious procedure in order to create a good program which achieved its intended goals (Dale & Weems, 2005). Lack of problem-solving skills and programming knowledge during the early stage will definitely interrupt the programming process from proceeding to the implementation phase. Thus, this will bring the programming process to a halt and reoccurring of iterative stage in the problem-solving phase. By understanding the programming phases, this will provide some guidance and insight to future researchers in developing and designing an effective content flow for teaching courseware.

Problem Solving Skills

In terms of problem-solving skills, students encounter a wide range of problem-solving skills. These include program design failure, confusion on the selection of loops, and inaccurate mental model.

Program design failure

The research done by Tan et al. (2009) further dwells into investigating the reasons behind the difficulties in learning of computer programming. Respondents agreed that they have problem designing a program to solve a certain task. This is due to the reason that they find it hard in dividing the functions into individual procedures which cause by the lack of problem-solving skills. This has proven that students face a lot of problems during the initial stage of understanding functions, procedures, and programs control. These problems exist long time ago and the finding corresponding with past research by (Bosse & Gerosa, 2017; Luxton-Reilly, 2016; Piwek & Savage, 2020; Savage & Piwek, 2019; Spohrer & Soloway, 1986). Novice programmers tend to merge different processes into a same piece of code when it should be implemented separately. Similar confusion happens when novices have the knowledge of the syntax (Bonar & Soloway, 1989; Robins, 2019; Tan et al., 2009) and semantics but fail to understand how to combine these features into a successful program (Winslow, 1996).

On another research done by Ismail et al. (2010b), findings show that the inadequacy of problem solving skills and capability of analysing a problem has contributed to the program design failure. This is due to the lack of understanding in programming concepts during the early stage of learning programming. The reason that causes this problem is the lack of prerequisite skills such as understanding of logic in programming course.

The problem further deteriorates when students are unable to understand and did not master the programming syntax and constructs well (Qian & Lehman, 2017). This can be detailed explain from the aspects of understanding the programming concepts, insufficient knowledge of syntax usage and unable to construct effective programming functions and procedures (Papadakis, Kalogiannakis, & Zaranis, 2016). Other researchers have also discovered the same reason as well (Aissa, Al-Kalbani, Al-Hatali, & BinTouq, 2020; Piwek & Savage, 2020; Robins, 2019).

In Malaysia, programming subject is made compulsory in the field of Information Technology (IT). It is a compulsory core subject that every IT and computer science student required to take in order to complete and graduate in an education program. However, the difficulties of teaching and learning of programming still remain unresolved. One of the main difficulties in learning of programming is to design a program to solve a certain task and dividing the functionality into procedures. Moreover, students encounter difficulties in learning the syntax of the language and eliminating bugs in the program which is time-consuming and tedious process (Qian & Lehman, 2017; Tan et al., 2009).

Confusion on the selection of loops

Knowing how to design a program and mastering the syntax of programming is just the beginning of a challenging task. According to the research conducted by Soloway, Bonar, and Ehrlich (1983), there are programming topics that cause lots of confusion pertaining to the usage of loops. The topic that causes a lot of confusion among the novice programmers is the “Program Control” which consists of selection structure and iteration structure. Besides that, various types of abstractions that implement the conditional statements produce many errors as the result of the earlier confusion (Hoc, 1984). Bugs and errors were commonly found in conditional statements and loops compared to other topics of programming (Robins, 2019; Spohrer & Soloway, 1986). In terms of the loops usage, Soloway et al. (1983) discover that novice programmers tend to opt for a “read then process” rather than a “process then read” strategies.

Past research conducted by Boulay (1986), do agrees that novice programmers are unable to understand the “for” loops. This is due to the updating of the loop control variable automatically “behind the scene” that requires the programmers to visualise the result. Besides that, another confusion that occurs similar to the hidden changes of variable automatically is the subscript value of an array (Robins, 2019; Savage & Piwek, 2019). All these processes require the programmer to perform the processing of data in their mental model

(Boulay, 2013). Existing knowledge, natural language, and analogies might also be the contributing factor misleading programmers when applied to the task in programming (Papadakis, Kalogiannakis, Orfanakis, & Zaranis, 2014). For example, based on natural language interpretation, novices assume that "while" loop will proceed continuously without being even tested once per iteration (Bonar & Soloway, 1985; Robins, 2019).

The problem worsens when misconception was discovered in other studies. This happens when novice programmers confused with the meaning of "while" loop in natural language. In programming terms "while" loop is typically used as a continually test in natural language. Further investigation revealed that even learning of algebra in constructing statement was also found to be confusing as well for the learners. For example, statement LET C = C + 1 did not make sense and considered to be an error. The researchers found that learners facing difficulties with understanding the concept of data input. As a result of that, the list continues as well for the concept "for-each" and "for-loop" (Kunkle & Allen, 2016).

Inaccurate mental model

The findings were further conceded by past researchers which refer to the mental model as "notional machines" (Boulay, 1986, 2013; Cañas, Bajo, & Gonzalvo, 1994; Du Boulay, O'Shea, & Monk, 1981). The purpose of "notional machines" is used to provide a basic understanding of the behaviour pertaining to a running program. This model is used as a basic abstract model for beginner to visualise and to relate some of the hidden effects. Besides that, it is also used by beginners to visually unmarked functions which cause problems for novice programmers to imagine. Further complications might happen as there are many ways of viewing a program in terms of the control flow, modular structure, linear order, and data flow (Rist, 1995; Robins, 2019). The most critical error that might occur is the projection of false mental model. Hence, this leads to incorrect design, wrong outcome and bugs which leads to the development of an incorrect end program. Minor bugs can be fixed without changing the program model whereas major bugs that required extensive adjustment might cause major conceptual changes.

Ineffective Pedagogy

In terms of pedagogy, the types of teaching materials, teaching strategies, and the content of the syllabus need to take into consideration. All these factors contribute to the success or failure in achieving a comprehensive learning environment.

Teaching materials

The usage of traditional teaching methods and static material such as books and slides does not seem to be an effective method. This is due to the reason that computer programming is a subject that is made up of dynamic nature concepts. Students fail to understand the dynamic nature of programs flow whenever static materials are used to explain the concepts to them (Gomes & Mendes, 2007; Zhang et al., 2019).

The Problem further persist when the usage of ineffective presentation techniques were implemented during the explanation of program flow. Conventional presentation techniques such as pseudo code and flow charts are used to explain problem solving method. The reason behind the failure is that conventional presentation techniques are ineffective in explaining the dynamic nature of computer programming. This conventional way is only suitable for teaching structured programming but not object-oriented programming (Gomes & Mendes, 2007).

Currently, most of the programming language learn are based on object-oriented programming approach. The teaching of object-oriented required more visualization to enable the student to have a mental representation of the problem (Robins, 2019). This problem links to the following contributing factors as well.

Teaching strategies

As mentioned earlier, current programming languages are mostly based on object-oriented programming. Thus, contemporary teaching strategies were no longer relevant and ineffective for problem solving method and coding. This is due to the paradigm shift of cognitive strategy in teaching of object-oriented

programming. The teaching of object-oriented programming requires the use of material that support mental model. Hence, the teaching materials which supported spatial and visualisation element are able to assist students in understanding the process of control and data flow (Savage & Piwek, 2019).

According to a research done at Multimedia University Malaysia by Tan et al. (2009), to determine types of learning methods which are preferred by the students in the university, it has shown that the students have rated learning by referring to programming examples has the highest rating. Other research has shown that learning by referring to programming examples achieved positive results as well (Brown & Wilson, 2018). While the second preferred method was the drill-practice. The research involved 182 undergraduates' respondents that is made up of 69.8% Malaysian and 30.2% International students. Besides that, interactive elements that consist of spatial and visualisations are much more useful than contemporary static programming material such as books and slides. Thus, learning using the contemporary approach would impair the learning process and further decrease learners' interest in learning programming.

In terms of the level of understanding in programming topics, a survey has also been conducted. The result shows that all the respondents have an average mean of less than three over five in all the programming topics which are below par. The most concerning issue is that it happens even at the beginning of the introduction of programming topics (Tan et al., 2009).

Syllabus

In terms of teaching syllabus, according to the research done by Soloway and Spohrer (2013), most of the introductory programming teaching material only emphasize on lower level knowledge such as declarative and procedural aspects. In other words, the students may know the syntax and semantics of programming but fail to combine them to create an effective computer program. Hence, it only emphasizes on 'what' and 'how' about programming concepts and syntax. This is only suitable for learning lower-level knowledge but not into solving complex programming problems in the real. Therefore, in the long-term students are unable to solve a programming problem as they are only equipped with low-level knowledge. These finding are further supported by the outcome obtained by past researchers (Bosse & Gerosa, 2017; McGill & Volet, 1997; Piwek & Savage, 2020; Rist, 1996; Savage & Piwek, 2019). Thus, it has shown that the problem persists since the 90's until the present time.

Besides that, instructors and lecturers are more focused on teaching the syntactic details of programming rather than promoting problem solving method. Moreover, the programming languages chosen are based on popularity in the industry rather than pedagogic purposes (Gomes & Mendes, 2007). Programming language should be chosen based on pedagogical aspect that benefits the learners. It should be easy to understand, remember, less complex, and self-explanatory to ease the learning cycle during the introduction stage.

Personal Traits and Attitude

It is undeniable that self-motivation and personal will of learning a subject sets the foundation for success. Students' personal traits such as motivation and interest do play an important role in making the learning process succeed. Hence analysis has been done in terms of personal traits and attitude. Three areas were identified, and they are the transfer of prior knowledge, negative perception, and attitude.

Transfer of prior knowledge

Personal traits and attitudes of students are key factors in contributing to the success or failure of learning programming subjects (Robins, 2019). The weak foundation of problem-solving skills during the initial stage leads to difficulty in interpreting the problem statement accurately. As a result, students failed to transfer prior knowledge and incorporated incorrect solutions. By implementing incorrect solutions, it proves that students' do not connect correct analogies with past problems (Gomes & Mendes, 2007). Moreover, lack of consolidation of prior knowledge capabilities in problem solving skills, and logic reasoning further worsens the problems during the early stage of learning (Bosse & Gerosa, 2017; Savage & Piwek, 2019).

Negative perception

In terms of belief and psychologically, students tend to have negative connotations relating to programming subjects. This is due to the reason of negative suggestions, opinions, and passing of comments from their senior peer which has taken the subjects before (Jenkins, 2002; Luxton-Reilly, 2016). Most of the comments given by their senior peer were negative which further diminished their interest. Moreover, the misconception on the difficulties in learning of computer programming during the early stage decreases students' motivation and confidence in excelling the subject (Qian & Lehman, 2017). Hence, bearing the negative impression and wrong perspective toward programming subjects leads the students to make their own conclusion that programming is an extremely difficult subject. It impaired the intrinsic motivation of the student which created a negative impact on their attitude to continue learning for success (Ng & Bereiter, 1991; Qian & Lehman, 2017; Rafique, Dou, Hussain, & Ahmed, 2020; Robins, 2019). However, there is research that shows using screencasting in teaching and learning of programming could increase the positive attitude towards the learning of programming (Cheah, 2019).

Attitude

Attitude plays an important role among the students in motivating and creating a positive impression in the computer programming subjects. It is imperative to have positive attitude as programming subjects required persistency and continuous learning from time to time (Robins, 2019). Students often ask for solutions or give up whenever they face obstacles. The perspective instills within the students are vital as it determines their intrinsic motivation. The level of motivation will reflect the amount of effort from the students whether to persist to strive or to capitulate in learning of computer programming (Qian & Lehman, 2017). Hence, it is necessary to ensure that a positive mind-set pertaining to computer programming subjects were cultivated from the initial stage of a programming course. Besides that, continuous learning support is needed to further instill positive impression. This will enhance and develop intrinsic motivation and positive attitude toward future programming subjects (Cheah, 2019; Jain & Sidhu, 2013; Qian & Lehman, 2017; Tai, Yu, Lai, & Lin, 2003).

CONCLUSION

Based on the plethora review of existing literature, there are various factors that contribute to the difficulties in teaching and learning of computer programming. Understanding the phases of programming stage is vital to ascertain whether the problems occur at the problem solving phase or implementation phase (Dale & Weems, 2013). It is crucial because within the phase, it consists of various types of skills and knowledge that need to be learned by the students. Moreover, the mastering of each knowledge and skill is necessary before proceeding to the following phase. As mentioned earlier, lack of understanding during the early stage of computer programming is one of the main factors contributing to the failure to understand computer programming (Bosse & Gerosa, 2017; Luxton-Reilly, 2016; Piwek & Savage, 2020; Savage & Piwek, 2019).

Traditional and conventional teaching methods using static material such as books and slides no longer relevant and effective in teaching and learning of computer programming (Gomes & Mendes, 2007; Zhang et al., 2019). Present programming languages consists of dynamic nature concepts which require teaching and learning tools that support spatial visualisation. Besides that, the programming language chosen should be based on pedagogic purpose rather that based on the popularity in the industry (Gomes & Mendes, 2007). This is to ensure that the students have the ease of learning during the early stage and have a strong foundation before moving into higher level of professional programming subjects.

In terms of learning methods, students preferred and learned more effectively by referring to programming examples (Tan et al., 2009). Syllabus can be arranged in a manner that focuses more on the difficult topics that are encountered by the majority of learners such as problem solving, functions, procedures, program control, and construct (Bonar & Soloway, 1989; Bosse & Gerosa, 2017; Boulay, 2013; Piwek & Savage, 2020; Rist, 1995; Savage & Piwek, 2019; Tan et al., 2009). Hence, multimedia elements can be incorporated into

teaching material such as screencasting and video to overcome the drawbacks of traditional teaching materials (Cheah, 2019).

Besides that, personal traits and attitudes play an important role in determining the intrinsic motivation. Negative personal belief and perception on the difficulties in learning of computer programming have contributed to the failure in learning of computer programming (Ng & Bereiter, 1991; Qian & Lehman, 2017; Rafique et al., 2020; Robins, 2019). This is due to the passing of comments, opinions, and suggestions from their senior peer which has taken the subjects before (Jenkins, 2002). Thus, educators should continuously instill positive perceptions and opinions during the early stage of programming education. Moreover, support should always be available during the foundation stage of learning computer programming to ease the learning process and instill confidence among the students (Qian & Lehman, 2017).

Therefore, a comprehensive teaching and learning material is needed to overcome the difficulties in learning computer programming. The teaching material should be able to support spatial visualisation by incorporating multimedia elements and interaction to explain the dynamic concept of programming (Cheah, 2019). Besides that, programming languages chosen should be based on pedagogic purposes, and the syllabus should be focused on the difficult topic encountered by most students. Furthermore, educators should always provide support and instill positive perception perceiving computer programming subjects to ease the learning process during the early stage of programming education. Future researchers can further explore into the cognitive aspect of learning computer programming to have a deeper understanding of the overall difficulties in the learning of computer programming.

SUMMARY

Findings done by past researchers (Gomes & Mendes, 2007; Soloway & Spohrer, 2013; Tan et al., 2009) have also proved that teaching and learning of programming indeed face lots of problems and difficulties. The outcome agrees with many other major findings which have been done by other researchers beginning 80's (Boulay, 1986; Davies, 1993; Du Boulay et al., 1981; McGill & Volet, 1997; Moser, 1997; Papadakis et al., 2016; Perkins, Hancock, Hobbs, Martin, & Simmons, 1986; Rist, 1995; Rogalski & Samurçay, 1990; Winslow, 1996). This indicates that the problem has been persisting for a long period of time. **Table 1** shows the contributing factors and its analysis that contribute to the difficulties in teaching and learning computer programming.

Table 1. Factors and Analysis Contributing to the Difficulties in Teaching and Learning of Computer Programming

Factors	Analysis
Phases of Programming Stage	- Problem Solving Phase - Implementation Phase
Problem Solving Skills	- Program Design Failure - Confusion on the Selection of Loops - Inaccurate Mental Model
Ineffective Pedagogy	- Teaching Materials - Teaching Strategies - Syllabus
Personal Traits and Attitude	- Transfer of Prior Knowledge - Negative Perception - Attitude

Therefore, the development of an effective teaching and learning material is important to overcome the weakness of novice programmers involving various types of factors mentioned above. In nature, computer programming is a highly complex subject to mastered and required substantial knowledge. Persistency, positive perception, and motivation are needed to overcome negative perceptions pertaining to difficulties in the subject. Besides that, effective application of strategies, comprehension of the overall process of programming, planning, and testing are necessary to eliminate unnecessary bugs and errors in the phases of

programming. Teaching material that would assist learners in developing the mental model and to understand how an executing program runs is one of the important element contain in using multimedia for education purposes. Finally, program control such as selection structure, and iteration structure have been identified as some of the most difficult topics for the novices to comprehend.

REFERENCES

Aissa, M., Al-Kalbani, M., Al-Hatali, S., & BinTouq, A. (2020). Novice Learning Programming Languages in Omani Higher Education Institution (Nizwa University) Issues, Challenges and Solutions *Sustainable Development and Social Responsibility—Volume 2* (pp. 143-148): Springer. https://doi.org/10.1007/978-3-030-32902-0_18

Baser, M. (2013). Attitude, Gender and Achievement in Computer Programming. *Online Submission*. https://doi.org/10.5829/idosi.mejsr.2013.14.2.2007

Bennedsen, J., & Caspersen, M. E. (2005). *Revealing the programming process.* Paper presented at the ACM SIGCSE Bulletin. https://doi.org/10.1145/1047344.1047413

Bonar, J., & Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human–Computer Interaction, 1*(2), 133-161. https://doi.org/10.1207/s15327051hci0102_3

Bonar, J., & Soloway, E. (1989). Preprogramming Knowledge: A Major Source of Misconceptions in Novice. https://doi.org/10.1207/s15327051hci0102_3

Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes, 41*(6), 1-6. https://doi.org/10.1145/3011286.3011301

Boulay, B. D. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research, 2*(1), 57-73. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

Boulay, B. D. (2013). Some difficulties of learning to program. *Studying the Novice Programmer*, 283. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

Brown, N. C., & Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS computational biology, 14*(4). https://doi.org/10.1371/journal.pcbi.1006023

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *SIGCSE Bull., 33*(3), 49-52. https://doi.org/10.1145/507758.377467

Cañas, J. J., Bajo, M. T., & Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies, 40*(5), 795-811. https://doi.org/10.1006/ijhc.1994.1038

Cheah, C. S. (2019). Screencasting: how effective is it in developing positive attitude towards the learning of C++ computer programming. *Journal of Educational Sciences & Psychology, 9*(2). Retrieved from http://jesp.upg-ploiesti.ro/index.php?option=com_phocadownload&view=file&id=553:screencasting-how-effective-is-it-in-developing-positive-attitude-towards-the-learning-of-c-computer-programming&Itemid=16

Chetty, J., & Barlow-Jones, G. (2014). Novice Students and Computer Programming: Toward Constructivist Pedagogy. *Mediterranean Journal of Social Sciences, 5*(14), 240. https://doi.org/10.5901/mjss.2014.v5n14p240

Chetty, J., & van der Westhuizen, D. (2014). *Implementing Metacognition Skills for Learners Studying Computer Programming.* Paper presented at the EdMedia: World Conference on Educational Media and Technology. Retrieved from https://www.learntechlib.org/primary/p/147775/

Dale, N., & Weems, C. (2005). *Programming and problem solving with C++*. Boston: Jones & Bartlett Learning.

Dale, N., & Weems, C. (2013). *Programming and Problem Solving with C++: Comprehensive*: Jones & Bartlett Publishers.

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies, 39*(2), 237-267. https://doi.org/10.1006/imms.1993.1061

Du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies, 14*(3), 237-249. https://doi.org/10.1016/S0020-7373(81)80056-9

Gomes, A., Carmo, L., Bigotte, E., & Mendes, A. (2006). *Mathematics and programming problem solving.* Paper presented at the 3rd E-Learning Conference–Computer Science Education. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.532.7543&rep=rep1&type=pdf

Gomes, A., & Mendes, A. J. (2007). *Learning to program-difficulties and solutions.* Paper presented at the International Conference on Engineering Education–ICEE. Retrieved from https://www.researchgate.net/profile/Anabela_Gomes2/publication/228328491_Learning_to_program_-_difficulties_and_solutions/links/02e7e52389017b9984000000.pdf

Hoc, J.-M. (1984). Do we really have conditional statements in our brains? *Readings on Cognitive Ergonomics—Mind and Computers* (pp. 92-101): Springer. https://doi.org/10.1007/3-540-13394-1_8

Holvikivi, J. (2010). Conditions for successful learning of programming skills *Key competencies in the knowledge society* (pp. 155-164): Springer. https://doi.org/10.1007/978-3-642-15378-5_15

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010a). 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, 42*(1), 35-61. https://doi.org/10.2190/EC.42.1.b

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010b). Instructional strategy in the teaching of computer programming: A need assessment analyses. *TOJET, 9*(2), 125-131. Retrieved from http://tojet.net/articles/v9i2/9214.pdf

Jain, Y., & Sidhu, G. K. (2013). Relationship between anxiety, attitude and motivation of tertiary students in learning English as a second language. *Procedia-Social and Behavioral Sciences, 90*, 114-123. Retrieved from https://doi.org/10.1016/j.sbspro.2013.07.072

Jenkins, T. (2002). *On the difficulty of learning to program.* Paper presented at the Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.596.9994&rep=rep1&type=pdf

Jiau, H. C., Chen, J. C., & Ssu, K.-F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4), 555-562. https://doi.org/10.1109/TE.2008.2010983

Kunkle, W. M., & Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE), 16*(1), 1-26. https://doi.org/10.1145/2785807

Luxton-Reilly, A. (2016). *Learning to program is easy.* Paper presented at the Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education. https://doi.org/10.1145/2899415.2899432

McGill, T. J., & Volet, S. E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of research on Computing in Education, 29*(3), 276-297. https://doi.org/10.1080/08886504.1997.10782199

Moser, R. (1997). *A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it.* Paper presented at the ACM SIGCSE Bulletin. https://doi.org/10.1145/268819.268853

Ng, E., & Bereiter, C. (1991). Three levels of goal orientation in learning. *Journal of the Learning Sciences, 1*(3-4), 243-271. https://doi.org/10.1080/10508406.1991.9671972

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2014). *Novice programming environments. Scratch & app inventor: a first comparison.* Paper presented at the Proceedings of the 2014 workshop on interaction design in educational environments. https://doi.org/10.1145/2643604.2643613

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organisation, 10*(3), 187-202. Retrieved from https://www.academia.edu/download/47316510/IJMLO10305_Papadakis_et_al.pdf

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research, 2*(1), 37-55. https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL

Piwek, P., & Savage, S. (2020). Challenges with Learning to Program and Problem Solve: An Analysis of Student Online Discussions. https://doi.org/10.1145/3328778.3366838

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE), 18*(1), 1-24. https://doi.org/10.1145/3077618

Rafique, W., Dou, W., Hussain, K., & Ahmed, K. (2020). Factors Influencing Programming Expertise in a Web-based E-learning Paradigm. *Online Learning, 24*(1). https://doi.org/10.24059/olj.v24i1.1956

Rist, R. S. (1995). Program structure and design. *Cognitive science, 19*(4), 507-562. https://doi.org/10.1207/s15516709cog1904_3

Rist, R. S. (1996). Teaching Eiffel as a first language. *Journal of Object-Oriented Programming, 9*(1), 30-41.

Robins, A. (2019). Novice programmers and introductory programming. *The Cambridge Handbook of Computing Education Research, Cambridge Handbooks in Psychology*, 327-376. Retrieved from https://www.otago.ac.nz/computer-science/otago706761.pdf

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Rogalski, J., & Samurçay, R. (1990). Acquisition of programming knowledge and skills. *Psychology of programming, 18*, 157-174. https://doi.org/10.1016/B978-0-12-350772-3.50015-X

Savage, S., & Piwek, P. (2019). Full report on challenges with learning to program and problem solve: an analysis of first year undergraduate Open University distance learning students' online discussions. Retrieved from http://oro.open.ac.uk/68073/1/IOC_TM112_Python_Help_Forum_Research.pdf

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM, 26*(11), 853-860. https://doi.org/10.1145/182.358436

Soloway, E., & Spohrer, J. C. (2013). *Studying the novice programmer*. Hillsdale, New Jersey: Psychology Press. https://doi.org/10.4324/9781315808321

Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM, 29*(7), 624-632. https://doi.org/10.1145/6138.6145

Tai, D. W., Yu, C.-H., Lai, L.-C., & Lin, S.-J. (2003). A study on the effects of spatial ability in promoting the logical thinking abilities of students with regard to programming language. *World Transactions on Engineering and Technology Education, 2*(2). Retrieved from http://www.wiete.com.au/journals/WTE&TE/Pages/Vol.2,%20No.2%20(2003)/Tai12.pdf

Tan, P.-H., Ting, C.-Y., & Ling, S.-W. (2009). *Learning difficulties in programming courses: undergraduates' perspective and perception.* Paper presented at the Computer Technology and Development, 2009. ICCTD'09. International Conference on. Retrieved from https://ieeexplore.ieee.org/abstract/document/5359977

Tüysüz, C. (2010). The effect of the virtual laboratory on students' achievement and attitude in chemistry. *International Online Journal of Educational Sciences, 2*(1), 37-53. Retrieved from https://www.academia.edu/download/55398446/Effect_Virtual_lab.pdf

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin, 28*(3), 17-22. https://doi.org/10.1145/234867.234872

Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2019). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education, 24*(2), 6. Retrieved from http://jise.org/Volume24/n2/JISEv24n2p147.pdf

Correspondence: Chin Soon Cheah, School of Educational Studies, Universiti Sains Malaysia, 11800, USM Penang, Malaysia. E-mail: ccs.cscheah@gmail.com