Functional Requirements of a C-Programming Problem-Solving Application

Nor Farahwahida Mohd Noor
Universiti Pendidikan Sultan Idris, Tanjung Malim, Perak
norfarahwahida@gmail.com

Aslina Saad
Universiti Pendidikan Sultan Idris, Tanjung Malim, Perak
aslina@fskik.upsi.edu.my

Azhani Hashim
Politeknik Sultan Azlan Shah, Behrang, Perak
azhani@psas.edu.my

Abstract
Problem-solving skill is very important to be inculcated among the students to produce graduates that meet the demand of the Industrial Revolution 4.0 (IR4.0). However, novice programmers often face difficulties to perform problem-solving before developing a computer program. Therefore, there is a need to develop an application that can assist programmers in problem-solving. It can be done with the application of computational thinking (CT) concepts. The purpose of the application is to assist novice programmers to solve programming problems, especially in C-programming. This study has elicited the functional requirements of a problem-solving application. Requirement elicitation is an initial software development process that determines the functional and non-functional requirements of the application. However, this paper focuses only on the application's functional requirements for the product requirement scope. The requirement elicitation was done using a triangulation strategy with qualitative approaches; semi-structured interview, document review, and an open-ended survey among five expert programming lecturers in Malaysian Polytechnics. The finding from this requirement elicitation has identified important elements that should be included in the problem-solving application. These elements are the input-process-output (IPO) chart which is combined with ten constructed scientific instructions and inquiries to facilitate the novices in problem-solving. The implication of these findings shows that the problem-solving process can be guided by the CT concepts to help students perform organized problem-solving and well-prepared programming coding.

Keywords: requirement elicitation, programming problem-solving, computational thinking.

1.0 Introduction

Industrial Revolution 4.0 (IR4.0) has driven a shift in industrial operations towards automation and artificial intelligence. This revolution has increased the demand for experts in the field of computer and software engineering, especially engineers and programmers. It also has changed industrial business models and employment trends, in which problem-solving skill has become the main criteria for employment.

Programming has been a challenging course for novices due to their lack of problem-solving skills. In programming learning, students need to have problem-solving skills based on computational thinking (CT) concepts.

CT simulates the thinking in the way the computer is processing. Therefore, the application of CT in problem-solving helps students in planning for the problem-solving in the way a program is executed. However, novices need a guide towards CT to plan the solution computationally so that they can clearly define the needs of the program coding. To cultivate CT among the novices, a problem-solving application needs to be developed. Therefore, the requirements of the application need to be clearly defined so that it manages to fulfil user needs towards the application.

This paper aims to elicit the functional requirements for a problem-solving application based on CT concepts. The application's purpose is to guide novices in performing problem-solving for basic scientific and engineering problems. It can be done by using the input-process-output (IPO) model with a set of scientific instructions and inquiries. This application is targeting students of an introductory programming course who are novice programmers. The intention is to help students plan for programming solutions computationally to perform problem-solving and understand programming better.

2.0 Literature review

Problem-solving is one of the most featuring skills needed for IR4.0 that needs to be trained among engineering students (Chaka, 2020). Moreover, this skill is one of the most sought by employers among graduate engineers to meet the demand of IR4.0 (Subramaniam et al., 2020). At higher learning institutions, problem-solving skill is essentially applied in the field of engineering and computer science study, which includes introductory programming courses (Malik et al., 2019). According to Chaka (2020), in learning programming, problem-solving is a predominant generic skill that is needed together with programming skills. However, the students often face difficulties in performing problem-solving (Hashim et al., 2017).

The lack of problem-solving skills is one of the main reasons that makes programming a challenging course for novices (Chung et al., 2016 & Veerasamy et al., 2019). However, problem-solving can be done systematically if the students have the skill to think in the way the computer is processing, which is known as the computational thinking (CT) skill (Moon et al., 2020; Svensson, 2020). CT is a universal skill that can be applied in solving problems involving programming in the aspects of task decomposition, abstraction, generalization, data structure and algorithm design (Riza et al., 2019). It is currently one of the top-level key skills that should be acquired to play in the industry (Saritepeci, 2020). Therefore, the CT concept should be applied to foster problem-solving skills among the students.

According to Kwon (2017), several common mistakes in programming were identified reflected from weak problem-solving strategies. During problem-solving, the novices often face difficulties in understanding the problem, planning solutions, designing and writing algorithms (Islam et al., 2019). Planning solutions is a difficult task for the novices if they are unable to decompose problems into functional data and procedures (Veerasamy et al., 2019). Therefore, as suggested by the CT concepts, the problem needs to be decomposed into manageable chunks to enable features extraction and abstraction of important data (Moon et al., 2020).

In programming, functional data are represented by variables. A clear understanding of the use of variables in programming is the basis of CT in solutions planning and algorithm design (Hosanee & Rana, 2018). However, the students tend to make mistakes that are identified mostly related to the misuse of variables in programming (Kwon, 2017). Therefore, a clear understanding of the role of variables to address data during problem-solving is very important to build good program coding.

Computationally, the variables are the elements in programming that address input and output data of a program with a descriptive name (Kohn, 2017). These variables can be determined if the students can extract important data and information from a problem. According to Svensson (2020), scientific instructions and inquiries can help to extract and identify important data and information from the problem.

The data and information from a problem can be categorized into input variables, processes, and output variables as described in the input-process-output (IPO) model. The IPO model is widely used in information processing for describing a process (Braunschweig, 2018). The combination of the IPO model and CT can help to guide students to express solutions in computational terms (Kwon, 2017). The use of the IPO model with the integration of CT in scientific instructions and inquiries are identified to be the elements that can assist problem-solving. Therefore, students will be able to clarify inputs, processes, and outputs of a problem if they are guided by scientific instructions and inquiries before they can proceed to design the algorithm of their program, and thus help them to program better.

In today's era of learning that applies the concept of e-learning, a problem-solving application is needed to help students solve problems more strategically and systematically. Therefore, requirement elicitation should be instigated as the first process to be done to develop this application (Supriya et al., 2018). In this process, the preparation of the application development is made, where the needs of the application are determined (Ramdhani et al., 2018). The needs of the application comprise important elements of problem-solving in programming which need to be defined to determine the product requirement. This product requirement should describe the goals of the application and the required features to achieve the goals. It also may include details about user interactivity with the functionalities and features.

3.0 Methodology

To determine the functional requirements of the problem-solving application, the requirement elicitation process is done by implementing the triangulation strategy. The triangulation strategy which combines several methods instead of only a single method is implemented to obtain high-quality requirements (Williamson, 2018). This strategy helps to acquire comprehensive findings where each method will complement the other (Saad & Dawson, 2018). In this strategy, several qualitative methods are applied to get an intensive analysis so that a detailed description of the desired result can be achieved (Schoch, 2020). These methods are semi-structured interviews, document review and open-ended surveys.

The requirement elicitation is carried out in three phases of these three different approaches. The research begins in Phase 1 with semi-structured

interview sessions with the expert programming lecturers. Then, it proceeds to Phase 2, which is the document review. After the documents are reviewed and some findings were made, the research proceeds to Phase 3, which involves an open-ended survey among the expert programming lecturers in Malaysian Polytechnics.

3.1 Phase 1: Semi-structured interview

The semi-structured interview is carried out to investigate the difficulties of the students in performing problem-solving for programming. This method is applied as it allows the researchers to explore subjective opinions and the expert's experiences in teaching programming (Evans, 2018). It aims to discover the role of CT and IPO Model in problem-solving from the view of experts programming lecturers. Besides, the researchers also intend to find out how instructions and inquiries could help novices in problem-solving. The interview topics are based on the literature review findings which have been discussed earlier. This interview helps the researchers to narrow down the scope in eliciting the functional requirements of the problem-solving application.

The interview is carried out among Malaysian Polytechnic lecturers who have been teaching the introductory programming course. Five expert programming lecturers were identified and voluntarily participated in this research. These lecturers are experience in teaching programming for more than 10 years. The semi-structured interview is chosen to allow the researcher to explore the respondents' feedbacks and inputs on the topics that have been outlined. The following are the topics that have been the guideline during the semi-structured interview:

i. The students' normal practice and difficulties in problem-solving.
ii. Elements that can assist students in performing problem-solving.
iii. IPO, CT, scientific instructions and inquiries application in problem-solving.

The first topic is to get a clear view and understand the current situation that usually happens during the problem-solving process in a programming practical session. This topic stimulates the respondents in suggesting several elements needed in the second topic of the interview. The second topic helps the researchers to define the elements of problem-solving, which could be the functional requirements of the application. The third topic focuses on getting inputs and feedback from the respondents if the elements suggested from the literature reviews are agreed to be applied as functional requirements of the application.

3.2 Phase 2: Document review

Document review and analysis are often used with other qualitative research methods in a triangulation strategy (Bowen, 2009). The purpose of the document review is to support the functional requirement that has been elicited during the interview. The document review is carried out to get an idea of the types of problems that are usually being used in programming practical activities. These practical activity documents need to be analysed to construct a set of instructions and inquiries as a problem-solving guide for these activities. This is supported by Bowen (2009) that documents may

contain information that can suggest some questions that need to be asked. Therefore, the researcher needs to compile an adequate amount of problem questions that are being used to practice problem-solving among novice programmers. The documents being reviewed are searched from practical lab sheets and assessment questions archives. The contents are limited to the fundamentals of programming, specifically the simple input-output and arithmetic functions. The review aims to analyse the problems to discover the suitable scientific instructions and inquiries that can be produced for each problem. In each of the problems, the inputs, outputs, and processes are identified to enable instructions and inquiries to be deduced. These inquiries are recorded and validated in the open-ended surveys.

3.3 Phase 3: Open-ended survey

The open-ended survey is done to validate the instructions and inquiries constructed for the application after the document review. These items are developed to be the general scientific instructions and inquiries which are expected to apply to most of the basic introductory programming problems. The purpose of the open-ended survey is to clarify if the items developed during the document review are valid and reliable, at the same time it also allows respondents to express comments and suggestions in their own words (Glazier et al., 2021). In the open-ended survey, the researcher will list down all the developed items to seek expert lecturers' consent. At each item, respondents are given some space to give their comments and suggestions. Other input items are also expected from the expert lecturers to produce a comprehensive set of instructions and inquiries that fit most programming problems.

4.0 Results and discussion

The requirement elicitation process has provided some information and led to the elicitation of the functional requirement for this problem-solving application. The findings are discussed according to each phase.

4.1 Semi-structured interview finding

The interview sessions were conducted separately among the expert lecturers. The interview finding of the students' normal practice and difficulties in problem-solving is summarized in Table 1. The table shows the responses of five respondents denoted by R1, R2, R3, R4 and R5. Five dominant practices (P1 – P5) from the responses have been identified as listed in the table. The check (√) symbol denotes that the item was from the corresponding respondent, meanwhile, the cross (X) symbol denotes that the item was not from the corresponding respondent.

Table 1: Interview findings of students' normal practice in problem-solving

No.	Students' Normal Practice	R1	R2	R3	R4	R5
P1	Search for codes from the internet to solve programming problems.	√	√	√	√	√
P2	Copy friends' answers.	√	√	√	√	√
P3	Modify code example.	√	√	√	√	√
P4	Extract data from problems.	√	X	X	√	X
P5	Produce the IPO chart before coding.	√	X	X	√	X

From this interview finding, the researchers have found some similarities in the respondents' responses. Based on the findings, the problem-solving process is skipped when the students try to copy codes from the internet or their friends. Modifying code examples also is not a part of the problem-solving steps that they need to undergo. The respondents were all agree that only a few students undergo the problem-solving steps by performing the solution planning and extracting out the IPO before coding the program. These findings show that the problem-solving skill among novice programmers is unsatisfactory. Therefore, the researcher concludes that CT is not well practiced among the novices which contribute to difficulties in the problem-solving process.

Table 2 summarizes eight suggested elements (E1-E8) by the five respondents for the application requirements to assist students in performing problem-solving. These elements are mostly in line with the suggested elements that the researchers already outlined for the third topic of the interview. Therefore, it also shows the accordance of the respondents to suggested elements of the literature reviews which are the CT, IPO and the use of instructions and inquiries. The table shows whether the element is agreed upon and suggested by the respondents denoted by the check (√) symbol. Meanwhile, the cross (X) symbol denotes that the respondents did not suggest the element.

Table 2: Suggested elements for a problem-solving application

No.	Problem-solving Elements	R1	R2	R3	R4	R5
E1	IPO chart to guide solution planning	√	√	√	√	√
E2	CT element in extracting data from problems	√	√	√	√	√
E3	Use inquiries and scientific instructions	√	√	√	√	√
E4	Identify input and output variables	√	√	√	√	√
E5	Identify involved processes and formulas	√	√	√	√	√
E6	Identify any conditional statement	X	√	√	X	√
E7	Identify any repetition statement	X	√	√	X	√
E8	Identify any conditional status	X	√	X	X	√

Based on this finding, the use of an IPO chart is agreed upon by all respondents as suggested by Kwon (2017). All respondents suggested that the use of an IPO chart be highlighted during the process to help in solution planning. All respondents also agreed with the implementation of CT in problem-solving to extract data from problems as suggested by Moon et al. (2020). They agreed that the extraction of data can be done by using scientific instructions and inquiries as suggested by Svensson (2020). These findings

are supported with additional suggestions from the respondents that the application must have the elements to help students identify input and output variables with the processes and formulas. Respondents R2, R3 and R5 also suggested the elements that help students with conditional and repetition statements. However, only two of them, which are R2 and R5 suggested proceeding with an element that helps identify the conditional status. Two respondents, R1 and R4 suggested that the application only focuses on basic arithmetic problem solving without conditional and repetition statements.

These findings show the importance of CT which is applied through instructions and inquiries to guide the novices in identifying the variables and related processes as parts of the task decomposition and abstraction processes. Based on the experts' responses, the researchers determined that seven elements out of eight (E1 – E7) will be the functional requirements of the problem-solving application. Element E8 is excluded as it serves beyond the basic elements of the introductory programming scope.

4.2 Document review finding

Twenty-five (25) problem-solving questions were compiled from various sources as shown in Table 3. These questions are basic scientific and engineering problems that require students to make a computer program to display certain outputs based on certain inputs after certain processes are done. To accomplish the task, the students need to discover the input and output variables from the problem. They need to identify how the output can be attained with scientific or arithmetic formula. These questions also require the novice programmer to identify suitable data types for each of the input and output variables that they have discovered.

Table 3: Sources of Problem-Solving Questions

No.	Problems Questions Sources	Number of Questions
1	Lab Sheets of Programming Fundamentals	10
2	Quiz Question	5
3	Test Question	5
4	Exercises	5

Based on suggested elements from the expert programming lecturer in Table 2, several instructions and inquiries (Q1-Q10) were developed from these sources. These instructions and inquiries are designed to extract data and information from the problems as shown in Table 4.

Table 4: General Instructions and Inquiries for Problem-Solving

No.	Instructions / Inquiries
Q1	What data will you get from the user? State the data type.
Q2	What should you calculate? State the data type.
Q3	What other data is needed? State the data type.
Q4	What other data is given? State the value. State the data type.
Q5	Give the formula to calculate the output.
Q6	Give a condition for a conditional action.
Q7	State an action if the condition is met.
Q8	State an action if the condition is not met.
Q9	Give a condition for a repetition action.
Q10	State a counter name if there is repetition.

These instructions and inquiries that have been constructed are validated in the third phase of the study by using an open-ended survey among the expert lecturers.

4.3 Open-ended survey finding

The open-ended survey was done to seek the expert programming lecturers' consensus with the instructions and inquiries (Q1-Q10) that have been developed in Table 4 based on their suggestions in Table 2.

Table 5 shows the respondents' consensus with instructions and inquiries in the open-ended survey denoted by the check (√) symbol. Meanwhile, the cross (X) symbol denotes that the respondent does not agree with the item.

Table 5: Experts' Consensus of the General Instructions and Inquiries for Problem-Solving

No.	Instructions / Inquiries	R1	R2	R3	R4	R5
Q1	What data will you get from the user? State the data type.	√	√	√	√	√
Q2	What should you calculate? State the data type.	√	√	√	√	√
Q3	What other data is needed? State the data type.	X	√	√	X	√
Q4	What other data is given? State the value. State the data type.	√	√	√	√	√
Q5	Give the formula to calculate the output.	√	√	√	√	√
Q6	Give a condition for a conditional action.	√	√	√	√	√
Q7	State an action if the condition is met.	√	√	√	√	√
Q8	State an action if the condition is not met.	X	√	√	√	√
Q9	Give a condition for a repetition action.	√	√	√	√	√
Q10	State a counter name if there is repetition.	√	√	√	√	√

Based on this open-ended survey finding, it is shown that all the instructions and inquiries items developed are agreed upon by the expert programming lecturers. Therefore, all the items are accepted to be included in the application. This finding shows the importance of abstracting important

data by identifying inputs, processes and outputs of the problem which is a part of the CT. However, some modifications need to be done based on some comments and suggestions. Table 6 summarizes the comments and suggestions (S1-S6) compiled from the open-ended survey among the expert programming lecturers. The check (√) symbol denotes the respondent who gives the suggestion and comment, while the NULL denotes that the respondent did not suggest the suggestion or comment.

Table 6: Suggestions and comments

No.	Instructions / Inquiries	R1	R2	R3	R4	R5
S1	Enable a reset function for IPO chart	-	√	√	-	√
S2	Give a clue or indication for every inquiry	√	-	√	√	√
S3	Instructions and inquiries are constructed as simple sentences.	√	-	-	√	√
S4	Q3 is combined with Q4 to reduce the number of questions	√	√	-	√	-
S5	Provide drop-down menu answer for a data type in Q1, Q2 and Q3	√	√	√	√	√
S6	Group the inquiries related to conditional statement and repetition statement respectively	√	√	√	√	√

Respondents R2, R3 and R5 suggested S1, enabling a reset function for element E1, which is the IPO chart to allow students to amend the solution that they have made. This suggestion is in line with the Semantic User Interface Guideline (SUIG) developed by Naidu & Saad (2018) which suggested that an application should help users to recognize, diagnose and recover from errors. SUIG also suggested that an application should allow user controls and freedom.

Respondents R1, R3, R4 and R5 suggested S2, giving a clue or indication for every instruction and inquiry so that the students can give a precise answer for each. The hints will help error prevention while giving inputs as suggested by the SUIG. Respondents R1, R4 and R5 suggested S3, that the instructions and inquiries are constructed in simple sentences. Long sentences with excessive information and designs should be avoided as they will confuse users (Naidu & Saad, 2018).

S4 is suggested by respondents R1, R2 and R4 to combine Q3 and Q4 to reduce the number of questions. All respondents suggested that the data type asked in Q1, Q2 and Q3 are given a drop-down menu answer form as stated in S5, so that the student can easily pick one answer from the available C programming data type. They also suggested S6, that the inquiries related to conditional statement and repetition statement are grouped, respectively. These suggestions S3, S4, S5 and S6 are very important to be considered to develop a simple application for the novices. They are in line with the SUIG that suggested maintaining an aesthetic and minimalist design with the efficiency of use.

5.0 Conclusion

The study was carried out to elicit the requirement of a problem-solving application, as part of the application development process. This application is targeting the introductory programming students who faced difficulties in problem-solving. The use of three different qualitative methods in the triangulation strategy has produced strong functional requirements for the application. Experts in programming learning emphasized that the use of the CT is crucial to guide the novices in problem-solving especially for the introductory programming course. The application should be able to guide users with the validated instructions and inquiries to identify input variables, related processes and output variables. It also should display these abstracted data in a form of an IPO chart to clarify the problem-solving process. Therefore, the problem-solving application will be developed with two important elements, which are the IPO Model and CT concepts represented in scientific instructions and inquiries to solve various basic problems in an introductory programming course.

6.0 Acknowledgment

The authors would like to acknowledge the Ministry of Higher Education for the support in conducting this research. This research is supported by Universiti Pendidikan Sultan Idris, Tanjung Malim, Perak, Malaysia. The authors also would like to express appreciation to the Malaysian Polytechnic and all the involved lecturers for supporting this research.

References

Bowen, G. A. (2009). Document analysis as a qualitative research method. Qualitative Research Journal, 9(2), 27–40. https://doi.org/10.3316/QRJ0902027

Chaka, C. (2020). Skills, Competencies and Literacies Attributed to 4IR/Industry 4.0: Scoping Review. *International Federation of Library Associations and Institutions*, *46*(4), 369–399. https://doi.org/10.1177/0340035219896376

Chung, I. L., Chou, C. M., Hsu, C. P., & Li, D. K. (2016). A programming learning diagnostic system using case-based reasoning method. *2016 IEEE International Conference on System Science and Engineering, ICSSE 2016*, 1–4. https://doi.org/10.1109/ICSSE.2016.7551544

Evans, C. (2018). Analysing Semi-Structured Interviews Using Thematic Analysis : Exploring Voluntary Civic Participation Analysing Semi-Structured Interviews Using Thematic Analysis : Exploring Voluntary Civic Participation. SAGE Research Methods Datasets.

Glazier, R. A., Boydstun, A. E., & Feezell, J. T. (2021). Self-coding: A method to assess semantic validity and bias when coding open-ended responses. Research and Politics, 8(3).

https://doi.org/10.1177/20531680211031752

Hashim, A. S., Ahmad, R., & Shahrul Amar, M. S. (2017). Difficulties in Learning Structured Programming: A Case Study in UTP. *Proceedings - 2017 7th World Engineering Education Forum, WEEF 2017- In Conjunction with 7th Regional Conference on Engineering Education and Research in Higher Education 2017, RCEE and RHEd 2017, 1st International STEAM Education Conference, STEAMEC 201*, 210–215. https://doi.org/10.1109/WEEF.2017.8467151

Islam, N., Shafi Sheikh, G., Fatima, R., & Alvi, F. (2019). A Study of Difficulties of Students in Learning Programming. *Journal of Education & Social Sciences*, *7*(2), 38–46. https://doi.org/10.20547/jess0721907203

Kohn, T. (2017). Variable evaluation: An exploration of novice programmers' understanding and common misconceptions. *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 345–350. https://doi.org/10.1145/3017680.3017724

Kwon, K. (2017). Novice Programmer's Misconception of Programming Reflected on Problem-Solving Plans. *International Journal of Computer Science Education in Schools*, *1*(4). https://doi.org/10.21585/ijcses.v1i4.19

Malik, S. I., Mathew, R., Al-Nuaimi, R., Al-Sideiri, A., & Coldwell-Neilson, J. (2019). Learning problem-solving skills: Comparison of E-learning and M-learning in an introductory programming course. *Education and Information Technologies*, *24*(5), 2779–2796. https://doi.org/10.1007/s10639-019-09896-1

Moon, J., Do, J., Lee, D., & Choi, G. W. (2020). A conceptual framework for teaching computational thinking in personalized OERs. *Smart Learning Environments*, *7*(1). https://doi.org/10.1186/s40561-019-0108-z

Naidu, T. J., & Saad, A. (2018). a Guideline for an Effective User Interface for Educational Semantic Application. *The International Journal of Multimedia & Its Applications*, *10*(06), 71–89. https://doi.org/10.5121/ijma.2018.10607

Ramdhani, M. A., Sa, D., Amin, A. S., & Aulawi, H. (2018). Requirements Elicitation in Software Engineering. *International Journal of Engineering & Technology*, *7*, 772–775.

Riza, L. S., Handoko, B., Wihardi, Y., & Herbert. (2019). Computational story: Learning media for algorithm and programming based on computational thinking. *International Journal of Engineering and Advanced Technology*, *9*(1), 2682–2685. https://doi.org/10.35940/ijeat.A9738.109119

Saad, A., & Dawson, C. (2018). Requirement elicitation techniques for an improved case-based lesson planning system. *Journal of Systems and Information Technology*, *20*(1), 19–32. https://doi.org/10.1108/JSIT-12-

2016-0080

Saritepeci, M. (2020). Developing Computational Thinking Skills of High School Students: Design-Based Learning Activities and Programming Tasks. *Asia-Pacific Education Researcher*, *29*(1), 35–54. https://doi.org/10.1007/s40299-019-00480-2

Schoch, K. (2020). Chapter 16 Case Study Research. In *The Scholar-Practitioner's Guide to Research Design* (pp. 245–256).

Subramaniam, M., Azmi, A. N., & Noordin, M. K. (2020). Problem Solving Skills Among Graduate Engineers: A Systematic Literature Review. *Journal of Computational and Theoretical Nanoscience*, *17*(2), 1044–1052. https://doi.org/10.1166/jctn.2020.8766

Supriya, P., Salve, M., Syed, P., Samreen, N., & Khatri-valmik, P. N. (2018). *A Comparative Study on Software Development Life Cycle Models*. 696–700.

Svensson, B. (2020). "Unplugged" Programming - A Way to Learn the Basics of Programming. *Proceedings of International Teacher Forum on International Conference on Computational Thinking Education 2020*, 21–22.

Veerasamy, A. K., D'Souza, D., Lindén, R., & Laakso, M. J. (2019). Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses. *Journal of Computer Assisted Learning*, *35*(2), 246–255. https://doi.org/10.1111/jcal.12326

Williamson, K. (2018). Chapter 13 Ethnographic research. In *Research Methods*. Elsevier Ltd. https://doi.org/10.1016/B978-0-08-102220-7.00013-3

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies*, *21*(3), 559–588. https://doi.org/10.1007/s10639-014-9341-9