JTH

https://jthkkss.com/
e-ISSN 2805-4431
DOI: https://doi.org/10.53797/jthkkss.v2i2.2.2021

eRequirement Elicitation Techniques for a C-Programming Learning Application

Mohd Noor, Nor Farahwahida & Saad, Aslina[1*]

[1]Sultan Idris Education University, 35900 Tanjong Malim, Perak, MALAYSIA

*Corresponding author email: aslina@fskik.upsi.edu.my

Available online 27 October 2021

Abstract: Requirement elicitation is a part of the application development process, which determines the functional and non-functional requirements of the application. This study has elicited the requirements of a programming learning application by using several requirement elicitation techniques. The purpose of the application is to assist novice programmers in learning C language programming. The requirement elicitation was done by using qualitative approaches in a triangulation strategy. The triangulation involved literature reviews on related existing programming applications and semi-structured interviews with five expert programming lecturers at Malaysian Polytechnic. The requirement elicitation has identified an important approach of a C-programming learning application: the programming visualization to help novices understand the program execution behaviour better. It has been determined that the application should visualize the variable contents and program execution steps as its functional requirements. Meanwhile, the application's non-functional requirement is that it should be designed to be a simple IDE-based application since it is targeting the novices. The finding from this requirement elicitation is essential in determining important functions to be available in the developed application and how it is supposed to be implemented. This application could enhance the students' programming skills and prepare them to be competent programmers for future industrial demands.

Keywords: Requirement elicitation, programming visualization, C programming

1. Introduction

Industrial Revolution 4.0 (IR4.0) has significantly impacted computing and software technology. It has significantly increased the demand for expert programmers (Kamaruzaman et al., 2019). Therefore, there is a need to produce computer science and engineering graduates with good programming skills (Chaka, 2020). However, learning programming is challenging for beginners (Cheah, 2020; Demilie, 2019). Many students claim it is difficult to understand the concept of programming and its execution behaviour (Islam et al., 2019; Qian & Lehman, 2019).

Programming needs competency in a specific programming language. However, the students, being novice programmers, are facing challenges to develop a program in syntactic, semantic and pragmatic aspects. These are reflected in their disfluency of writing in the language syntax, which results in unresolved syntax errors (Ettles et al., 2018).

They also need help understanding the program's behaviour due to their inability to visualize how it works (Hashim et al., 2017). This results in poor programming skills among novices. Furthermore, using a professional Integrated Development Environment (IDE), overwhelmed with complex functions, to demonstrate programming is daunting for novices (Warner & Guo, 2017).

Many studies have been done to identify various approaches to tackle these issues. One of the approaches being applied is programming visualization. Programming visualization is a method to show a computer program's internal structure and behaviour that cannot be physically visualized (Shin, 2018). In the research context, programming visualization is the construction of visual images of the program. These visual images show the current state of program execution steps and the result of the program execution on variables. The result of the program execution can be shown by using graphical representations that show the contents of the variables. This means that programming visualization is envisioning the memory contents related to the program.

Corresponding author: aslina@fskik.upsi.edu.my

There have been several existing programming visualization applications being developed for programming learning. However, they mostly target other programming languages, but comparatively few are in the C-language (Egan & Mcdonald, 2020). Therefore, this research will focus on enhancing programming learning in the C-language among the novices. Realizing the benefits of programming visualization for the novices, a programming learning application that employs programming visualization in the C-language should be developed.

To develop this application, the first process that should be instigated is the requirement elicitation of the application (Salve et al., 2018). The requirement elicitation is a preparation phase process where the users' needs are determined (Ramdhani et al., 2018). The requirement elicitation process can be done using several methods rather than only applying a single method. The combination of several requirement elicitation methods is known as the triangulation strategy (Williamson, 2018). Triangulation helps develop high-quality requirements as each technique will complement each other to support the multidimensionality of the requirements investigations (Saad & Dawson, 2018).

This paper aims to elicit the application requirements and acquire knowledge of the approaches in helping novices in learning programming, to develop a C-programming learning application. The application will employ the programming visualization concept. The application could help students understand programming better by visualizing the variables' contents within the program memory and program execution steps using a simple graphical presentation.

2. Methodology

The requirement elicitation process used qualitative approaches with a triangulation strategy. Qualitative approaches help the researcher to get a detailed description and intensive analysis of the related issues in this research (Schoch, 2020). The approaches implemented in this research were a literature review and semi-structured interviews. The requirement elicitation was carried out in two phases: Phase 1 (One) is the literature review and Phase 2 (Two) involves semi-structured interviews with the expert programming lecturer.

2.1 Phase 1: Literature Review

A literature review was carried out to investigate the programming visualization approaches employed in programming learning. This method was chosen as it allows researchers to further analyze primary research findings to provide new interpretations and summarized knowledge of a specific topic (Lubbe et al., 2020). All existing programming visualization applications available in the C language were identified in this method. These applications were analyzed and compared. The features and qualities of these applications were then recorded. Among the features being recorded were the application platforms, design approach, compiler availability, ability to visualize variable contents, execution steps, extra supporting features, and the main drawbacks of each. The analysis of these applications was compiled and used during the semi-structured interview sessions.

2.2 Phase 2: Semi-structured Interview

The semi-structured interview was conducted to support and clarify the literature review finding, thus contributing to the application requirement elicitation. This method was applied to explore subjective opinions based on the respondents' experiences (Evans & Lewis, 2018). The interview was carried out among programming lecturers in the Malaysian Polytechnic. Five expert programming lecturers were identified for the semi-structured interview sessions. They were chosen for having qualifications in the subject area with more than 10 years of teaching experience (Wolff et al., 2021). With that teaching experience, they possess the criteria of an expert lecturer for being involved in developing teaching materials and assessments and addressing different types of learners (Reyes & Torio, 2021). Besides, they also have experience using different types of IDEs and programming applications in teaching programming.

The semi-structured interviews were done with basic guidelines for defining the application requirements. These guidelines helped the researcher focus on helping students learn to program using visualization. The following three topic themes were the guidelines of the interview sessions: 1) the students' difficulties in writing C-programs; and 2) what features of programming visualization should be developed; and 3) how should the programming visualization application be applied.

The first was the ice-breaking theme to introduce the research subject matter to the respondents. It also helped the researcher to understand the current situation in programming learning. The second and third themes were to elicit the functional and non-functional requirements, respectively.

The researcher used the teach-back technique to support the second and third topics in the interview sessions. Using this technique, the compilation of several programming visualization applications was shown to the expert lecturers. This technique helped the researcher clarify the suitable approaches based on the existing applications (Roslan et al., 2021). During the interview sessions, the researcher sought their feedback on the suggested approaches from the applications that have been reviewed. Besides, other relevant inputs from the lecturers were also expected to get better approaches to help their students learn to program better.

3. Results

The requirement elicitation techniques have brought some important findings. These findings led to the elicitation of the programming learning application's functional and non-functional requirements. The application's functional requirement is very important as it determines what to be designed and developed. Meanwhile, the non-functional requirement determines how the application should be designed and developed.

3.1 Literature Review Results

The literature review has discovered six related programming visualization applications available in the C-language. These applications are the Python Tutor (PT), SeeC (SC), PlayVisualizerC (PVC), Visual Interpreter (VIP), Flash and Scratch. These applications are analyzed and summarized as shown in Table 1. Based on Table 1, the analysis was made mainly by comparing the application design approach and the drawbacks of each application, which could be improved in this study.

Table 1: Analysis of programming visualization applications

Applications / Features aspect	PT	SC	PVC	VIP	Flash	Scratch
Platform	Web-based	Standalone	Web-based	Web-based & standalone	Web-based	Web-based & standalone
Design approach	IDE	IDE	IDE	IDE	Animation	IDE
Compiler	GCC	Clang	Not available	CUP & JFlex	Not available	Yes
Visualize variable content	Yes	Yes	Yes	Yes	Yes	Not available
Visualize execution steps	Yes	Yes	Yes	Yes	Yes	Not available
Supporting features	Supports other programming languages	Support run-time error detection	Can visualize the relationship of variables to memory	Includes expression evaluation	No need to develop a program	Drag-and-drop block-coding
Drawbacks	Does not support standard input/output in C	Difficult set-up environment, cannot modify source code	It does not compile; thus, no error message is produced	Needs a webserver to run the application's profiler	Cannot demonstrate their own program	No syntax learning

3.2 Semi-structured Interview Result

The teach-back technique has clarified the needs of the application to be developed. Through the semi-structured interview, the researcher managed to acquire the main elements to be visualized in the application, which are the functional requirements of the application.

Table 2 summarizes the semi-structured interview finding among the expert programming lecturers. Based on the interview theme, the respondents have identified eight requirements of the application (F1-F8) based on programming visualization. These requirements have been suggested and supported by at least four out of five respondents. These respondents are those five lecturers R1, R2, R3, R4 and R5 denote. The table shows the respondents' agreement with the suggested functions and features of the proposed application, denoted by the check (√) symbol. Meanwhile, the cross (X) symbol denotes that the respondent thinks the proposed function or feature is not crucial.

Table 2: Analysis of semi-structured interviews among expert lecturers

No	The features of visualization that need to be present	R1	R2	R3	R4	R5
F1	Highlight changes in variable content on each line of program implementation	√	√	√	√	√
F2	The variables' contents must be updated at every single instruction execution	√	X	√	√	√
F3	Should have a graphical representation of	√	√	√	√	√

Table 2: Analysis of semi-structured interviews among expert lectures (Continued)

No	The features of visualization that need to be present	R1	R2	R3	R4	R5
	the variables' contents					
F4	Should show the program execution steps	√	√	√	√	√
F5	The application points out the current command line being executed, as well as the previously executed command line	√	√	√	√	X
F6	The application should be developed as an IDE-based application to allow students to demonstrate their own program	√	√	X	√	√
F7	The application should be implemented as a standalone application rather than as a web-based application	X	√	√	√	√
F8	The application must be presented as simple as possible by using graphical illustrations	√	√	√	√	√

4. Discussion

Application design approach as shown in Table 1, from these six applications, Scratch IDE is an attractive application for novices that implements visual programming. However, this approach is not suitable to be applied for the learning of C-language syntax (Bakar et al., 2019). The visual programming approach applied in Scratch does not support the transition to text-based programming since it emphasizes building programs using block-based programming (Sim & Lau, 2018). However, its drag-and-drop block-coding feature is helpful to assist novices in programming which could be implemented for future research.

Meanwhile, the Flash application was another unique approach that implemented animation rather than an IDE approach. Being an animation application, it does not function as an IDE. By only observing the animation of the programming execution process, students need to be more engaging in active learning. Students should be learning programming by the learning-by-doing concept through IDE, which could enhance active learning (Gallego-Romero et al., 2020). Moreover, using an animation-based application in programming learning is unsuitable because it does not allow the students to demonstrate their program within the application.

On the other hand, the PT, SC, PVC and VIP are found to be IDE-based applications. Compilers support PT, SC, and VIP, while the PVC uses an interpreter for program interpretation. Therefore, these applications allow students to actively demonstrate their program and acquire programming experience. However, some drawbacks in these applications open up room for improvement that is attempted in this study.

Furthermore, the drawbacks of the existing applications, although PT, SC, PVC and VIP are very much supporting programming learning, there are some drawbacks of these applications. As has been documented by the web administrator, the PT does not support the input-output in the C language. Specifically, it does not support using the scanf () function, the basic function often used by students to give input to the program. It is an essential feature of the C language in the course syllabus (Department of Polytechnic and Community College Education, 2019). This limitation has made this application less usable for C language fundamentals learning.

Meanwhile, although the SC supports the C language input-output, the SC is difficult to install because it needs a compiler setup before the application installation. This complex installation process could increase the extraneous cognitive load for the novices. Moreover, a program source code cannot be modified within the SC environment, complicating the learning process.

Reviewing the next application, it was found that the PVC application has no compiler. Therefore, it cannot compile a program made by the student and produce an error message if there is any. This has been a major drawback of the application because the ability of the application to compile a C program is a benefit in learning to program.

Most of the applications were built to be web-based applications. However, this will cause a significant dependency on internet connection. This will imply bandwidth cost and performance issues for its dependencies on internet connectivity and speed. This matter should be considered, which could affect the students' readiness (Azhar & Rani, 2020).

Although the SC and VIP applications are available in the standalone version, several drawbacks make these applications not applicable to novices. Although VIP is available as both web-based and standalone versions, the standalone version still needs a webserver to run the application's profiler. Meanwhile, the SC application has a complex environment before users can use it. The SC is dependent on the Clang compiler, which complicates installation and offline environment setup (Ishizue et al., 2020). The advantages and disadvantages of the application that has been reviewed were referenced during the interview with the expert programming lecturers.

Meanwhile, based on the data in Table 2, all the expert programming lecturers agreed with F1 that programming visualization should highlight variable content changes on each program implementation line. They also agreed with F2 that the variables' contents must be updated at every instruction execution. Therefore, they suggested F3 that the

application should have a graphical representation of the variables' contents. Apart from that, the respondents suggested F4 that the application show the program execution steps. Additionally, they suggested F5 that the application point out the current command line being executed and the previously executed command line. This would help the students to analyze how the program is executed. These five requirements (F1-F5) have been identified as the application's functional requirements.

The interview also clarified F6 that the application should be developed as an IDE-based application. This is to allow students to demonstrate their program. Therefore, the application should be supported by a suitable compiler. Based on the respondents' responses on F7, the application is preferred to be implemented as a standalone rather than a web-based application. A standalone application will allow students to use the application without needing an internet connection (Dumbiri & Nwadiani, 2020). The lecturers also suggested F8 that the application be presented as simply as possible using graphical illustrations. These three later requirements (F6-F8) have been identified as the non-functional requirements of the application.

5. Conclusion

The study was carried out to elicit a programming learning application development requirement. The learning application targets the C language. Using the triangulation strategy during the requirement elicitation has resulted in strong and mutually supportive findings. The expert lecturers agree that programming visualization is the best approach to be applied in the application. It will be visualizing two main elements: the variable contents and the program execution steps. It should be graphically presented in a simple IDE-based environment to make programming look easy for the novices. For future studies, since visual programming has become an interesting approach for beginners, a visual programming approach can assist in programming at a higher education level. Text-based programming can be adapted to block-based programming through frame-based programming. Whatever the approach, syntax learning should not be at stake because syntax is the foundation of programming and remains an industry practice.

Acknowledgement

The authors want to acknowledge the Ministry of Higher Education's scholarship support in conducting this research. Sultan Idris Education University, Tanjung Malim, Perak, Malaysia, support this research. The authors would like to express appreciation to Sultan Idris Education University for the valuable comments and other bits of help. The authors also would like to express appreciation to the Head of the Electrical Engineering Department, the Head of the Program, Diploma in Electronic (Computer) Engineering and all programming lecturers in Polytechnic Sultan Azlan Shah, Behrang Perak and other Malaysian Polytechnics who have supported and involved in this research.

References

Azhar, N., & Rani, N. C. A. (2020). Student Readiness Towards E-Learning Adoption in Higher Education: A Conceptual Model Based on Extended Technology Acceptance Model. *Journal on Technical and Vocational Education*, *5*(2), 61-74. *Scribbr*. http://upikpolimas.edu.my/ojs/index.php/JTVE/article/view/382

Bakar, M. A., Mukhtar, M., & Khalid, F. (2019). The development of a visual output approach for programming via the application of cognitive load theory and constructivism. *International Journal of Advanced Computer Science and Applications*, *10*(11), 305-312. https://doi.org/10.14569/IJACSA.2019.0101142

Chaka, C. (2020). Skills, competencies and literacies attributed to 4IR/Industry 4.0: Scoping review. *IFLA Journal*, *46*(4), 369-399. https://doi.org/10.1177/0340035219896376

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, *12*(2), 1-14. https://doi.org/10.30935/cedtech/8247

Demilie, W. B. (2019). Causes of failure of university students in computer programming courses: The case of Wachemo University. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *5*(5), 123-132. https://doi.org/10.32628/cseit195516

Department of Polytechnic and Community College Education. (2019). *Course Information DEC20012 Programming Fundamentals.*

Dumbiri, D. N., & Nwadiani, C. O. (2020). Challenges Facing Application of E-learning Facilities in Vocational and Technical Education Program in South Nigeria Universities. *Asian Journal of Vocational Education and Humanities*, *1*(2), 1-8. https://doi.org/10.53797/ajvah.v1i2.1.2020

Egan, M. H, & McDonald, C. (2021). An evaluation of SeeC: a tool designed to assist novice C programmers with program understanding and debugging. *Computer Science Education*, *31*(3), 340-373. https://doi.org/10.1080/08993408.2020.1777034

Ettles, A., Luxton-Reilly, A., & Denny, P. (2018, January). Common logic errors made by novice programmers. In *Proceedings of the 20th Australasian Computing Education Conference* (pp. 83-89). https://doi.org/10.1145/3160489.3160493

Evans, C., & Lewis, J. (2018). *Analysing semi-structured interviews using thematic analysis: Exploring voluntary civic participation among adults*. London. SAGE Publication Limited.

Gallego-Romero, J. M., Alario-Hoyos, C., Estévez-Ayres, I., & Delgado Kloos, C. (2020). Analyzing learners' engagement and behavior in MOOCs on programming with the Codeboard IDE. *Educational Technology Research and Development*, *68*, 2505-2528. https://doi.org/10.1007/s11423-020-09773-6

Hashim, A. S., Ahmad, R., & Amar, M. S. S. (2017). Difficulties in Learning Structured Programming: A Case Study in UTP. In *2017 7th World Engineering Education Forum (WEEF)*, 210-215. IEEE. https://doi.org/10.1109/WEEF.2017.8467151

Ishizue, R., Sakamoto, K., Washizaki, H., & Fukazawa, Y. (2020). PVC. js: Visualizing C programs on web browsers for novices. *Heliyon*, *6*(4), 1-15. https://doi.org/10.1016/j.heliyon.2020.e03806

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

Kamaruzaman, F. M., Hamid, R., Mutalib, A. A., & Rasul, M. S. (2019). Conceptual framework for the development of 4IR skills for engineering graduates. *Global Journal of Engineering Education*, *21*(1), 54-61.

Lubbe, W., ten Ham-Baloyi, W., & Smit, K. (2020). The integrative literature review as a research method: A demonstration review of research on neurodevelopmental supportive care in preterm infants. *Journal of Neonatal Nursing*, *26*(6), 308-315. https://doi.org/10.1016/j.jnn.2020.04.006

Qian, Y., & Lehman, J. D. (2019). Using targeted feedback to address common student misconceptions in introductory programming: A data-driven approach. *SAGE Open*, *9*(4), 1-12. https://doi.org/10.1177/2158244019885136

Ramdhani, M. A., Maylawati, D. S. A., Amin, A. S., & Aulawi, H. (2018). Requirements elicitation in software engineering. *International Journal of Engineering & Technology*, *7*(2.19), 772-775. *Scribbr*. https://etheses.uinsgd.ac.id/10580/

Delos Reyes, R. D. G., & Torio, V. A. G. (2021). The relationship of expert teacher–learner rapport and learner autonomy in the CVIF-dynamic learning program. *The Asia-Pacific Education Researcher*, *30*(5), 471-481. https://doi.org/10.1007/s40299-020-00532-y

Roslan, R., Ayub, A. F. M., Ghazali, N., & Zulkifli, N. N. (2021). The development of a collaborated gamified e-quiz and strategy game mobile application to increase students' motivation and continuance usage intention. *ANP Journal of Social Science and Humanities*, *2*(2), 74-81. https://doi.org/10.53797/anp.jssh.v2i2.10.2021

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

Salve, S. M., Samreen, S. N., & Khatri-Valmik, N. (2018). A Comparative Study on Software Development Life Cycle Models. *International Research Journal of Engineering and Technology*, *5*(2), 696-700. *Scribbr*. https://www.irjet.net/archives/V5/i2/IRJET-V5I2154.pdf

Schoch, K. (2020). Case study research. *Research design and methods: An applied guide for the scholar-practitioner*, 245-258.

Shin, W. (2018). A Study on the Effects of Visualization Tools on Debugging Program and Extending Functionality. *International Journal of Advanced Science and Technology*, *115*, 149-160. https://doi.org/http://dx.doi.org/10.14257/ijast.2018.115.14

Sim, T. Y., & Lau, S. L. (2018, November). Online tools to support novice programming: A systematic review. In *2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, 91-96. IEEE. https://doi.org/10.1109/IC3e.2018.8632649

Warner, J., & Guo, P. J. (2017, May). Codepilot: Scaffolding end-to-end collaborative software development for novice programmers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (pp. 1136-1141). https://doi.org/10.1145/3025453.3025876

Williamson, K. (2018). *Chapter 13: Ethnographic research. Research Methods (2nd Ed.), Information, Systems, and Contexts*, 311-335. https://doi.org/10.1016/B978-0-08-102220-7.00013-3

Wolff, C. E., Jarodzka, H., & Boshuizen, H. P. (2021). Classroom management scripts: A theoretical model contrasting

expert and novice teachers' knowledge and awareness of classroom events. *Educational Psychology Review*, *33*, 131-148. https://doi.org/10.1007/s10648-020-09542-0