AN EDUCATIONAL ESCAPE ROOM FOR COMPUTATIONAL THINKING - DEFINING THE REQUIREMENTS

Peter Mozelius[1], Lisa Sällvin[1], & Niklas Humble[2]
[1]Mid Sweden University (Sweden)
[2]University of Gävle (Sweden)

Abstract

This paper presents an educational development project where game-based learning is used to facilitate introductory programming courses in higher education. The identified problem that is addressed in the project is the low pass rate and low student satisfaction in university courses on fundamental programming. A recommended pre-training for programming is computational thinking, and to learn about the fundamental concepts that are involved in programming, independent of specific programming languages. An initial literary review revealed that there exist several educational games on the combination of computational thinking and programming, However, these games are targeted towards a younger target group, or that they have a focus on specific programming. The aim of this study is to explicate the described problem, and to gather requirements for the design and development of an educational escape room. The research project follows the design science approach where the first two steps of 1) explicate the problem and 2) define the requirements were studied and described in this paper. The problem to address in the study was identified through literature searches and the authors' experiences as teachers in programming at higher education. To address the identified problem, requirements for a digital game were defined through e-mail interviews with teachers in higher education that teach fundamental programming courses. Answers were collected from teachers from three different universities in Sweden and analysed with open coding. Findings identified through the analysis will be used in future research studies to address the remaining steps of the design science methodology and further iterations of development. Findings show that some fundamental concepts seem to be relatively easy to introduce while others are harder to grasp for students taking their first programming course. Examples of concepts that could be learnt relatively easy are variables and non-nested selection. Some concepts that are seen as harder to introduce and explain are nested iteration and ternary operators. The conclusion is to build a game with different levels of thematic escape rooms, where the first levels have a focus on what teachers mentioned as easy concepts. The highest levels should introduce the more complex concepts, but that the concepts that are seen as most problematic could be omitted. This study was the first iteration in the definition of requirements, and more interviews will be conducted and analysed in the next phase of this two-year project.

Keywords: *Game based learning, educational escape room, computational thinking, fundamental programming, design science.*

---

1. Introduction

Programming education at university level is classified as problematic learning with low pass rates and high drop-out rates for introductory courses on the fundamentals of programming (Gomes & Mendes, 2007; Cheah, 2020). Many first-year students on undergraduate programmes in Computer Science and Informatics fear the introductory programming courses more than other courses (Gomes & Mendes, 2007; Watson & Li, 2014; Lukose, 2021). A frequently recommended pre-training for programming is computational thinking (CT), building on the idea learning about the fundamental concepts of programming before starting out with actual coding (Lyon & Magana, 2020)). CT should, by definition, be independent of specific programming languages, but involving concepts that are useful in concrete programming. Another popular way of motivating students is game-based learning, where games for learning computer science and programming could be implemented as educational escape rooms (Borrego et al., 2017; López-Pernas et al, 2019).

This study is a part of the SPEDAT project, where an earlier literary review revealed that there exist several educational games on computational thinking and programming, However, the conclusion was that these games are targeted towards a younger audience, or that the focus is on specific programming languages. The aim of this study is to explicate the described problem, and to gather requirements for the design and development of an educational escape room. These two research steps are part of a two-year project plan where the final delivery will be an educational escape room to learn CT. The research questions that guided this study were:

RQ1: Which CT or programming concepts do teachers find easy for students to learn, and which concepts are hard to learn?

RQ2: Which CT concepts or programming concepts do teachers find to be important pre-knowledge for programming courses, and how should they be implemented in an educational escape room?

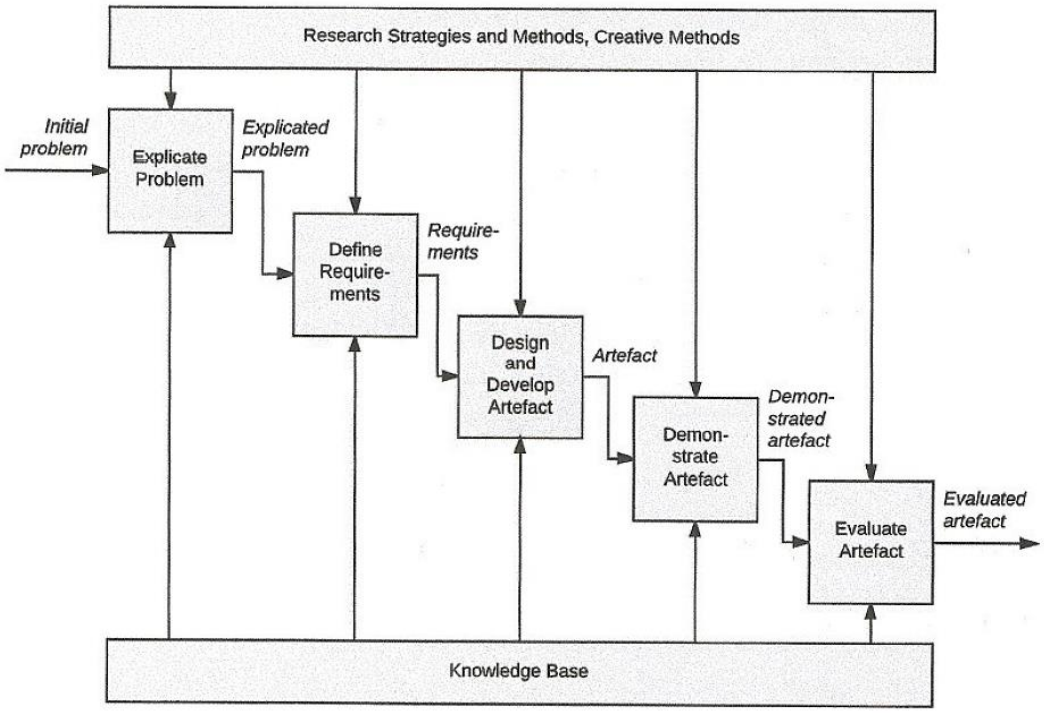
2. Research context

This study is a part of the second phase of the SPEDAT project, a two-year educational development project carried out in a collaboration between the Mid Sweden University, and University of Gävle. SPEDAT is a Swedish acronym that could be translated to English as 'Games for computational thinking'. The aim of the SPEDAT project is too design and develop an educational escape room game where the players can learn about computational thinking and programming, without alignment to any specific programming language. In the second part of the project the main focus is on explicating the underlying problem and to gather requirements for the design and the development of the escape room game. The study has followed the Design science research approach that is described in the next section.

3. Method

This study was conducted with a Design Science approach involving the two first phases of the five-phase process that has been outlined by Johannesson and Perjons (2014). The first two phases that were studied and described in this paper were 1) To explicate the problem and 2) To define the requirements. All five phases in the Design Science process are depicted in Figure 1 here below.

Figure 1. The five phases in the Design science framework (Johannesson & Perjons, 2014, p. 82).



To strive for quality requirements, phase two should be carried out iteratively with the data collection and the analyses organised as described in the subsections below.

3.1. Data collection

The explication of the problem in phase 1 was based on the combination of results from literature searches, and from authors' earlier experiences as teachers in programming courses at university level. To address the identified problem, and to gather requirements for an educational game e-mail interviews were sent to teachers in higher education with experiences from programming courses. In this first iteration of the requirement definition answers were collected from six teachers from three different universities in Sweden. Informants were selected with the idea of a purposive expert sampling (Rai & Thapa, 2015), consisting of teachers that all have long and rich experience of teaching programming in higher education. All data were gathered during the end of 2022 and in the beginning of 2023. Later during 2023 more e-mail interviews will be conducted and analysed with the aim of data saturation. All informants have been kept as anonymous as possible during the process.

3.2. Data analysis

This first analysis phase was conducted by two of the authors following the Grounded theory concept of open coding as described by Khandkar (2009). In an inductive open coding process, researchers started by fracturing data into discrete parts, and thoroughly, by close reading, examine the parts to identify data extracts, codes and preliminary categories. In the next step that Khandkar (2009) refers to as 'Abstracting the concepts', data (the e-mail interviews) were divided into distinct and labelled ideas, events or objects. The name of these labels can be decided by the involved analysers, be derived from the content, so called 'in vivo codes'. During the analysis process, it can be difficult to describe all concepts in just a few words, which in open coding is complemented with explaining descriptions or 'memos' (Glaser & Strauss, 1967; Khandkar, 2009). Finally, the abstracted concepts are reanalysed and organised into categories, before writing up and presenting the findings. Open coding has also been called initial coding, and the first of three thematic analyses in a process that also should involve axial coding, and selective coding (Morris et al., 2016).

4. Findings and discussion

The six informants are all experienced programming teachers with between 7 to +25 years of teaching at university level. The most frequently used programming languages among these informants are Python, Java, C++ and JavaScript. Following the principle of answering the research questions, results from the analysis have been grouped into the four categories of 'Easy-to-Learn Concepts', 'Hard-to-Learn Concepts', 'Important prerequisites', and 'General game design'. These four categories are presented and discussed one by one here below. The findings in the first and the second category answers RQ1, while the findings in the third and the fourth category addresses RQ2.

4.1. Easy-to-learn concepts

One of the informants wrote that "variables, printing and isolated if-then-else statements seem to be easier to understand", and another answer mentions that "variables, constants, assignment ant print-outs" are what students understand easily. Several other interview answers also bring up variables, if-clauses, print-outs and simple mathematical expressions. In one interview the teacher finds it difficult to answer which concepts students in general learn easily since "There are students in our distance courses that completes the whole course without asking about anything, while others ask many questions about most parts of the course without learning how to program". The student groups in introductory programming course are often heterogeneous, and maybe with greater differences regarding pre-knowledge than in other subjects. Students that have earlier experiences of programming, which could be rather superficial would probably not find variables, constants and print-out statements hard to understand. This is also one of several reasons for developing educational games on computational thinking and fundamental programming. With just a bit of understanding of how the basic concepts work, the first programming courses at university level, would probably be a less painful experience. Finally, a quote from an informant with students that must have more than good pre-requisites: "To translate from programming language A to B do they find amusing, and that they also learn from it, despite that it is easy. It's like when your musical understanding develops when you transpose a song from one key to another".

4.2. Hard-to-learn concepts

One of the six informants contradicts the consensus on variables as an easy-to-learn concept. It was claimed in this answer that "The concept variable use to be a threshold. You have the habit of using variables in mathematics, but then as a numeric value that could be calculated in an equation. To instead interpret it as a place for storing values that can be changed, as in a programming language, goes against earlier experience". The other threshold that was pointed out in the same answer, the concept of functions, or methods, or procedures, or sub-routines, is also part of most of the other answers. Independent of earlier use of functions in mathematics, the way they are used in programming is puzzling with calls, recursive calls, parameters and return values. The Hard-to-Learn Concept that appears in most interview answers is object-orientation, and that the design of classes and creation of objects seem abstract to many students. A bit depending on the used programming language, object-orientation can be omitted in the introductory courses and the same must be the conclusion for our educational escape room. On the other hand, an idea to keep for the future might be the one about another specific game on object-oriented concepts only. Other brought up Hard-to-Learn Concepts were nested selection and nested iteration, about which a teacher wrote that: "Even the basics with while-loops gets them confused, nested loops or if statements, and ternary operator. Many students (specially the ones that already start the course knowing programming) use for-loops with break instead of while / do-while loops". Moreover, the informants mentioned that it is hard for novice programmers to follow the execution flow in programs, and to search for errors in code with complex flow control.

4.3. Important prerequisites

There are many concepts on the informants wishlists, some of them involves object-orientation, but most concepts refer to imperative or procedural programming. What appears to be interesting concepts to implement in our educational game are: algorithmic thinking, problem solving, basic computer knowledge such as file handling and naming conventions, and visualisation of algorithms and data structures. A suggestion for training of algorithmic thinking and visualisation of algorithms was "Which well-defined steps would it take to solve a problem, and in what order should they be executed? As an example, create an algorithm that moves the game protagonist further on, or kills an antagonist". The same informant suggested that this should be implemented as "black boxes that could be seen from both the inside and the outside". Several good suggestions, but we would rather like to call the transparent boxes 'glass boxes', and to keep the game free from violence. An important principle for the authors is to go for inclusive design, and not to scare off girls with unmotivated violence (Mozelius et al., 2022b). All suggestions are worth considering for the game even if some of them seems to advanced, for players without earlier programming experience, like the one on "to follow and implement a simple API". There are also suggestions that are to language specific such as the one on: " exploring more complex data types such as vector<int> in C++". Furthermore, the belief that we find highly relevant is "I believe that the improvement of computational thinking into designing algorithms (seeing logical sequences of steps). Also, they should learn the fundamentals of different statements to create algorithms (what, how and why we use: variables, input, printing, control flow, etc.)". Concepts that all would fit in well in the gameplay that we have sketched upon in the projects early brainstorming sessions.

4.4. General game design

Regarding the general game design, one informant's recommendations for pair programming are valid also for the single player game that we are planning for. In formal and informal learning, instructors could group the students, to play the game together while exchanging thoughts. The game could also be complemented with a 'play & study guide' with ideas for pre- and post-play activities. In non-formal learning this could be self-organised in the same way as many other single players games are played by more than one student. The recommendations implementing leader boards based on points should be considered, but the idea of giving points for "good-looking code" does not fit into the programming language independent design decision. One informant has the more general wish of "A learning game with programming - where programming is used for, yes, playing", and also with the recommendation of having a look at Sonic Pi (sonic-pi.net). Other informants recommend to get inspiration from other existing games such as the problem-solving game 'Opus Magnum', and the Zachtronic puzzle games that involves programming concepts. Several informants also highlight the idea of a number increasingly challenging levels in the game, and that also the protagonist should develop during the game play. Finally, a relevant idea from one of the interviewees is that each level, or escape room, should be linked to information that could help the players to escape from the actual room. This idea could be implanted as a variation of the game-based learning concept called 'Tangential learning' (Mozelius, Fagerström & Söderquist, 2017).

5. Conclusion

The conclusion is to design and develop an educational game with increasingly complex levels of thematic escape rooms. In the first levels the focus should be on what teachers mentioned as easy concepts. The highest levels should introduce the more advanced concepts, but that the concepts that the informants saw as most the most problematic could be omitted. This study should be seen as the first iteration in the definition of requirements, more interviews with programming teachers will be conducted and analysed in the next phase of the SPEDAT project. However, based on the findings in this study, the educational game should not involve any object-oriented concepts. This would make the desired playful encounter with computational thinking more complex and less joyful, and that the learning of 'Object-oriented thinking' better should be done in other specific educational games.

6. Future work

This initial open coding analysis should in the next iteration be followed up by an axial coding where data should be reanalysed to revise the categories presented in this paper. Moreover, axial coding is an analytic process that investigates interrelationships between the categories that earlier have been developed in the open coding process (Strauss & Corbin, 1990).

Acknowledgement

This study was facilitated by funding from the Higher Education and Digitalisation (HEaD) initiative, a five-year strategic initiative at the Mid Sweden University, described in detail in (Mozelius et al., 2022a). The HEaD project aims to improve the university capacity in the field of lifelong and technology enhanced learning. HEaD do not only focus on the development of teachers' pedagogical digital competence, but also has the ambition to improve the support structure available to teachers. SPEDAT is one of several two-year educational development projects in the second round of the HEaD initiative.

References

Borrego, C., Fernández, C., Blanes, I., & Robles, S. (2017). Room escape at class: Escape games activities to facilitate the motivation and learning in computer science. *JOTSE*, *7*(2), 162-171.

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, *12*(2), p 272.

Gomes, A., & Mendes, A. J. (2007). Learning to program – Difficulties and solutions. In International Conference on Engineering Education – ICEE (Vol. 2007).

Johannesson, P., & Perjons, E. (2014). *An introduction to design science*. Cham: Springer.

Khandkar, S. H. (2009). Open coding. *University of Calgary, 23*, 2009.

Lukose, J. M. (2021). Effects of Guided Inquiry-based Learning and Teaching Approach on Students' Confidence, Motivation and Communication Skills in an Introductory Computer Programming Course. *Multicultural Education*, *7*(12).

Lyon, J. A., & J. Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*, *28*(5), 1174-1189.

Morris, M. T., Daluiski, A., & Dy, C. J. (2016). A thematic analysis of online discussion boards for brachial plexus injury. *The Journal of Hand Surgery*, *41*(8), 813-818.

Mozelius, P., Fagerström, A., & Söderquist, M. (2017). Motivating Factors and Tangential Learning for Knowledge Acquisition in Educational Games. *Electronic Journal of e-Learning*, *15*(4), 343-354.

Mozelius, P., Bader, S., Jaldemark, J., Urbansson, P., & Engström, A. (2022a). Educational Development-Challenges, Opportunities, Tools and Techniques. In European Conference on e-Learning (Vol. 21, No. 1, pp. 264-271).

Mozelius, P., Humble, N., Sällvin, L., Öberg, L. M., Pechuel, R., & Fernández-Manjón, B. (2022b). How to get the girls Gaming: A Literature Study on Inclusive Design. In 16th European Conference on Game Based Learning (ECGBL 2022).

Rai, N., & Thapa, B. (2015). A study on purposive sampling method in research. *Kathmandu: Kathmandu School of Law*, 1–12.

Strauss, A., & Corbin, J. (1990). *Basics of qualitative research: Grounded theory procedures and techniques*. Sage.

Watson, C., & Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 39-44).