Mapping on the Use of Games for Programming Teaching with an Emphasis on Software Reuse

Diego Castro, Cláudia Werner

COPPE/Computer Systems Engineering Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
{diegocbcastro, werner}@cos.ufrj.br

Abstract—**Many works have already approached the use of games as a teaching method due to several advantages that this strategy can bring to the current teaching method. Therefore, a study was previously performed to identify games created for teaching software reuse fundamentals. However, no work addressing this problem was identified. Software reuse is an essential area of software engineering and is commonly associated with programming. Based on this, this article sought to identify works that had already been done on the use of games for programming teaching but could be used to teach reuse fundamentals.**

Keywords–game; game-based-learning; software reuse; programming; systematic mapping.

I. Introduction

Software Reuse (SR) is the discipline responsible for creating software systems from pre-existing software [1]. This concept is not just limited to code reuse; software in this context can refer to other products, such as modeling, specifications, test plans, and any other product in the life cycle of a project. With the correct use of this discipline, it can provide several positive impacts in a variety of contexts, such as quality, cost, productivity, code-making performance, rapid prototyping, reduced code writing, reliability, and interoperability of software [2].

Despite the advantages mentioned, SR is still far from being used on a large scale; many people reuse software, but not in a systematic manner. One of the main factors for reuse not being implemented is the difficulty of education in the area [3]. Based on this, a study was previously performed to identify games created for teaching software reuse fundamentals. However, it was not possible to identify a game specifically designed for this purpose.

In this initial research, it was observed that software reuse might be contained in different areas, such as programming and Software Engineering. Thus, this initial research was divided into two parts: the first to search for games to teach software engineering and the other to search for games for programming teaching with an emphasis on software reuse. Based on the information provided, this study aims to identify games that have the purpose of teaching programming with emphasis/potential for reuse, that is, to find games that were developed for teaching programming, but could be used to teach some of the fundamentals of software reuse, such as logical reasoning development, function development, object orientation, among others.

The remainder of this paper is presented as follows: Section II describes the research method used in the systematic mapping, Section III shows some results that were found, Section IV demonstrates an example of how one of the games found could be used to teach SR, Section **??** shows the threats to validity, and Section V concludes with the final remarks.

II. Research Method

Systematic mapping is a secondary study method based on a structured and repeatable process or protocol that explores studies and provides a result in the form of an overview of a particular subject [4]. The mapping presented follows the protocol proposed by Kitchenham [5].

The research process presented in this study covers articles published by the end of 2018 and aims to conduct a systematic mapping to identify work that has already been done on games for programming teaching but could be used to teach software reuse fundamentals, such as logical reasoning development, function development, object orientation, among others.

A. Research Questions

- **Q1:** What is the main advantage / motivation of the use of games to teaching programming language?
- **Q2:** What is the disadvantage of the use of games to teaching programming language?
- **Q3:** What is the main characteristic of the game used?
- **Q4:** What was the evaluation method used?

The mapping presented followed well-defined steps so that it was possible to reach a set of articles that were of interest to the search [5]. The search string was executed in Scopus as recommended by other studies [6] [7], and then the inclusion and exclusion criteria were applied to the set of articles that were found based on the title, abstract, and full text.

The **inclusion criteria** chosen were: **(1)** The article must be in the context of using games for teaching a programming language; **(2)** The article must provide data to answer at least one of the research questions; **(3)** The article should be written in English. The **exclusion criteria** chosen were: **(1)** Book chapters, conference call; **(2)** Studies that can not be fully accessed.

B. Search string and Analysis

The definition of the search string was based on the **P**opulation, **I**ntervention, **C**omparison, **O**utcome (PICO) structure [8], using three of the four levels. The search string was defined by grouping the keywords of the same domain with the logical operator "OR" and grouping the two fields with the logical operator "AND". However, we chose to use a date filter, searching only for articles that were published within five years, aiming to find more recent works in the area [9]. Table I demonstrates the PICO structure used in conjunction with the search string.

Initially, the search string returned a total of 507 papers. When analyzed according to the inclusion and exclusion filters, this number dropped to 17 papers. To minimize the lack of other search bases, considering that the study was only performed on Scopus, it was opted to use the snowballing procedure to minimize article loss, according to Motta et al. [6] and Matalonga et al. [7]. The approach was applied, searching for new papers through the references and through the papers that referenced these works. Thus, 9 more papers were included, totaling 26 analyzed papers. Table II shows how these 26 articles were obtained, and Table III shows each of these articles.

TABLE I. SEARCH STRING

PICO	SYNONYMS
Population	Programming language, algorithm experience, algorithm skills, algorithm alternative, algorithm method, coding experience, coding skills, coding method, coding alternative
Intervention	Tutoring, teach*,instruction, discipline, schooling, education*, mentoring, course, learn*,train*, syllabus
Comparison	Not applicable
Outcome	Game*, gami*, play*, "serious games", edutainment, "game based learning", simulation
SEARCH STRING	
TITLE-ABS-KEY (("programming language" OR "algorithm experience" OR "algorithm skills" OR "algorithm alternative" OR "algorithm method" OR "coding experience" OR "coding skills" OR "coding method" OR "coding alternative") AND (tutoring OR teach* OR instruction OR discipline OR schooling OR educat* OR mentoring OR course OR learn* OR train* OR syllabus) AND (game* OR play* OR "serious games" OR gami* OR edutainment) AND (LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016) OR LIMIT-TO (PUBYEAR , 2015) OR LIMIT-TO (PUBYEAR , 2014)))	

III. RESULTS

Section A demonstrates a discussion of the main results found in this work, and Section B presents the threats to the validity of this information exposed.

A. Discussion

The articles found in this study sought to demonstrate games that could be used in teaching some concepts related to programming. However, the analysis of the documents was performed in search of works that could be used to explain some of the concepts of reuse. From this, works that were not developed with this context but could be used for this purpose were also found. Figure 1 groups the articles by location and year of publication. It is possible to see an increase in the number of publications over the years and that many countries are looking for improvements in this area.

The bottom of the image also shows the number of articles found grouped by game type. However, some papers used more than one approach. From Figure 1, it is possible to observe that the most used way to teach programming is through the use of "blocks of code". By abstracting this idea it is possible to consider the concept of software components that is an important research area of SR and aims to build software from pre-produced components [10].

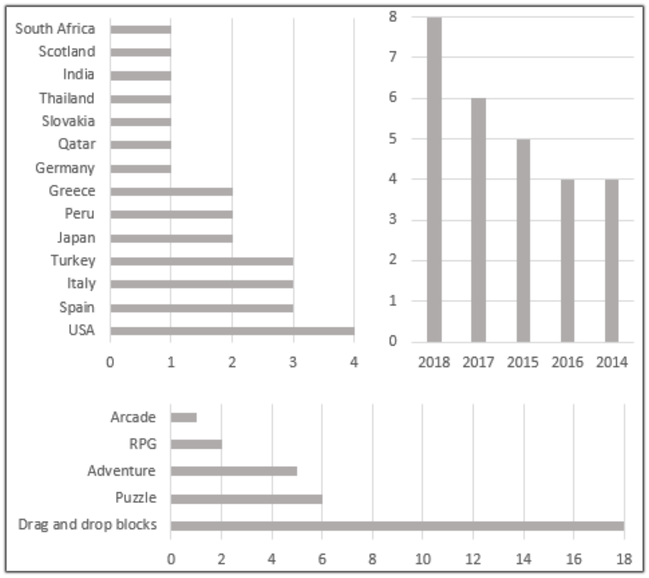

Figure 1. General analysis of the articles found.

Q1: What is the main advantage / motivation of the use of games to teaching programming language?

Using games as a reinforcement tool to teach skills can be a very beneficial strategy for students. They have proven to be a useful tool to complement conventional learning methods. Games allow visualizing concepts that may be too abstract. They also help you get acquainted with the knowledge and methods that may be tedious to study, offering a cycle of challenges and rewards that drives the learning experience [11].

Many authors claim that games have several characteristics that can benefit teaching [12] [13]. They have already been used as successful educational tools in many different fields and topics, such as engineering, learning languages, theater and even health [14]. The advantages include: increased student motivation and engagement, enhancement of pre-existing knowledge, increased performance in practical activities, immediate feedback, fun and satisfaction, among others [11] [15–19].

Q2: What is the disadvantage of the use of games to teaching programming language?

Despite the advantages offered by games as a teaching method, there are also some issues involving this approach. The first problem found was the comparison of the level of learning provided by a game as a teaching method and a class with textual programming. Despite the advantages offered by games, textual programming can still convey better content [20].

Another problem identified was the complexity of the game created. If the teaching tool used is too complicated, students can reduce the time spent solving problems to focus more on the tool. This is an unwanted distraction, and any game used should be easy to use, allowing the student to focus on solving the problem rather than how to use the game [21].

Finally, the last problem identified was that although games provide several advantages, they are not seen as self-sufficient.

TABLE II. ANALYSIS OF THE PAPERS

	Main Study		Snowballing backward		Snowballing Forward	
Activity	Result	Number of paper	Result	Number of paper	Result	Number of paper
First Execution	507 added	507	389 added	389	123 added	123
Repeated Papers	501 withdraw	501	294 withdraw	95	16 withdraw	107
Papers in another language	0 withdraw	501	14 withdraw	81	13 withdraw	94
Remove conference / workshops	16 withdraw	485	0 withdraw	81	0 withdraw	94
Remove books	0 withdraw	485	0 withdraw	81	0 withdraw	94
Remove by title	368 withdraw	117	46 withdraw	35	58 withdraw	36
Remove by abstract	83 withdraw	34	17 withdraw	18	18 withdraw	18
Papers not found	0 withdraw	34	0 withdraw	18	0 withdraw	18
Remove by full paper	17 withdraw	17	12 withdraw	6	13 withdraw	3
Total papers included	17 papers		6 papers		3 papers	
Extracted Papers	26 papers					

TABLE III. TRACEABILITY MATRIX.

Title	Year	Q1	Q2	Q3	Q4
Perceptions of Scratch programming among secondary school students in KwaZulu-Natal, South Africa	2018	X		X	X
Robo3: A Puzzle Game to Learn Coding	2018	X	X	X	X
Improving programming skills in engineering education through problem-based game projects with Scratch	2018	X		X	X
Introducing novice programmers to functions and recursion using computer games	2018	X		X	X
Introducing programming using “scratch” and “greenfoot”	2018	X		X	X
Developing Educational 3D Games With StarLogo: The Role of Backwards Fading in the Transfer of Programming Experience	2018	X	X	X	X
Learning to think and practice computationally via a 3D simulation game	2018	X		X	X
Design and implementation of Robo3 : an applied game for teaching introductory programming	2017	X		X	X
A cross-cultural review of lightbot for introducing functions and code reuse	2017	X		X	
Using Digital Game as Compiler to Motivate C Programming Language Learning in Higher Education	2017	X		X	X
Cubely: Virtual reality block-based programming environment	2017	X		X	X
Analysis of the learning effects between text-based and visual-based beginner programming environments	2017	X		X	X
Visual programming language for model checkers based on google blockly	2017	X		X	X
Educational resource based on games for the reinforcement of engineering learning programming in mobile devices	2016	X		X	X
Teaching abstraction, function and reuse in the first class of CS1 - A lightbot experience	2016	X		X	X
From Alice to Python Introducing text-based programming in middle schools	2016	X		X	X
Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch” in five schools	2016	X		X	X
Building a Scalable Game Engine to Teach Computer Science Languages	2015	X		X	X
A mobile-device based serious gaming approach for teaching and learning Java programming	2015	X		X	
Coding with Scratch: The design of an educational setting for Elementary pre-service teachers	2015	X		X	X
Droplet, a Blocks-based Editor for Text Code	2015	X		X	X
Integrating Droplet into Applab – Improving the usability of a blocks-based text edit	2015	X		X	X
The development of a virtual learning platform for teaching concurrent programming languages in secondary education: The use of open Sim and Scratch4OS	2014	X	X	X	X
Effects of using Alice and Scratch in an introductory programming course for corrective instruction	2014	X		X	X
A structured approach to teaching recursion using cargo-bot	2014	X		X	X
The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners, Perspective, Informatics in Education	2014	X		X	X

Professional follow-up and feedback on the course are required to solve any problem that may arise throughout the learning process [21].

Q3: What is the main characteristic of the game used?

This study identified several games that sought to teach programming through increased motivation and engagement through fun. Most of these games were designed to be used by users with minimal or no knowledge of programming language [22].

The first game found was LightBot, which is a game to teach programming logic and has features such as multi-level, difficulty progression, feedback, challenges, use of similar tasks, concepts of functions, abstraction, flow control, recursion and code reuse [15] [16] [19]. Another game very similar to the one described above is Cargo-Bot, which has the same characteristics, but with other gameplay that revolves around a crane that moves and stacks a set of colored boxes. Players write small programs to move boxes from one initial setup to another [23]. Another game called Robo3 was found that had characteristics very similar to those described [11].

Another game very similar to the ones listed above was a game designed to teach Java programming that, to advance the levels, the player needs to overcome different levels. As the player surpasses these levels, he or she can progress through the story, unlocking new elements and gaining experience points to unlock new content [22].

Another game found was Lost in Space, which includes, among other components, a game rules system, a physics engine, and a rendering engine. The game screen is divided into two parts. The left side containing the code interpreter text area and a help window and on the other side, the game phase. Through this game, some features were highlighted, such as obstacles, code interpreter (pseudocode of the game), collisions, movement, enemy and attack system [14].

In this research, we also identified some visual programming languages that are not considered as games directly, but that uses ”block” approach to building programs. The first two to be identified were Alice [24] and Scratch [17], which are block-based visual programming languages designed to promote media manipulation for new programmers. From these languages, it is possible to upload media projects and scripts, animated stories, games, book reports, greeting cards, music videos, tutorials, simulations, and art and music projects. Two other languages very similar to those described are StarLogo TNG [20] and Droplet [25] [26], which are also drag-and-drop visual languages.

Greenfoot is an integrated tool that aims to teach object-oriented programming. Also, the tool allows teachers to introduce the most essential and fundamental concepts of object orientation in an easily understandable way [27]. Finally, the last visual language found is called Google Blockly [28], which is a library for building visual programming editors.

Finally, another feature that was used to create these games was the use of virtual and augmented reality. The Cubely game made use of these technologies to develop an idea that blended block programming concepts and the Minecraft game [29].

Q4: What was the evaluation method used?

Several evaluation methods were identified in this research. However, in general, all evaluations have a questionnaire applied to a specific population after using the tool to validate it [20] [24] [28].

Another possible means of the evaluation was the use of control groups where one group used the tool, and the other did not, and the same questionnaire was applied to both groups [14]. Through this assessment, it is possible to find out if there was a gain of experience through the tool use since it is possible to compare the results of the two groups.

The last evaluation method found was about the use of the tool as part of the discipline — the tool as a complement to the teaching of programming [30].

To conclude, games can be a new method to complement the current teaching method due to its main advantages, such as increased practice and engagement through challenges, rewards, fun, and feedback. However, it is still something new that needs attention due to problems such as the complexity of the game that can affect learning, and the level of learning provided by games is still lower than current teaching methods.

B. Threats to Validity

Through a critical analysis of the mapping, it is possible to perceive some threats that may have affected the final result of the work. The first to be highlighted is about the period in which the mapping was performed, collecting information from just five years. The second threat is the problem of interpreting the information found, which is up to the author to understand the game found and think of a way that could be applied in the teaching of SR.

IV. Reusing games to teach SR

This mapping found several games; however, none of them was produced to teach SR. Nevertheless, these games, with only a few or no modifications, could be used to explain certain concepts of software reuse, such as the importance of reusing, software components or code reuse.

Thinking about this idea of teaching SR, the platforms of Scratch, Alice, Droplet, and Google Blockly could, for example, be used to teach code reuse. All code that is generated with these platform is made from pre-produced blocks that resemble the idea of pre-produced components.

Robo3 and Lightbot are of puzzle type and are very similar, the general idea of these games is to create sequences of activities (which are described as functions) that perform a task, such as taking the avatar from point A to point B. Thinking about this type of games, these functions can be used in the game several times, teaching the student the concept of code reuse. Cubely is a Minecraft-based game; however, its mechanics are very similar to the two games described earlier and could also be used to teach code reuse.

V. Conclusion and Future Work

For many people who are not directly linked to the software reuse area, they refer to it as just code. Due to this fact, this mapping sought to find programming teaching games that could be used to teach reuse concepts that are often abstract to many students. From this, it was possible to identify six games and six block-based programming languages. The game, and the visual programming language that were identified in more articles were LightBot [15] and Scratch [17], respectively. The main characteristics found were the use of rules, phases, difficult progression, feedback, challenges, and the use of similar tasks in sequence.

As mentioned before, software reuse is inserted in several contexts, and the most common are propagation and engineering. This work sought to identify games that were created to teach programming but could be used to explain some of the fundamentals of software reuse, thus looking at works from the first context. To better understand how these games are used as teaching methods, it is intended to perform another mapping to identify games that aim to teach software engineering, since as software reuse is inserted in the engineering and possibly similar features can be used to the teaching of the two subjects.

Although this work has found some games that could be used to teach some reuse fundamentals such as components, functions, and object orientation, none of these games were specificaly designed to teach software reuse. Therefore, based on the characteristics that were found (multi-level, difficult progression, feedback, challenges, among others), it is intended to create a game with the specific purpose of software reuse teaching.

REFERENCES

[1] C. W. Krueger, "Software reuse," ACM Computing Surveys (CSUR), vol. 24, no. 2, 1992, pp. 131–183.

[2] J. Sametinger, Software engineering with reusable components. Springer Science & Business Media, 1997.

[3] N. Niu, D. Reese, K. Xie, and C. Smith, "Reuse a" software reuse" course," in American Society for Engineering Education. American Society for Engineering Education, 2011.

[4] I. Steinmacher, A. P. Chaves, and M. A. Gerosa, "Awareness support in distributed software development: A systematic review and mapping of the literature," Computer Supported Cooperative Work (CSCW), vol. 22, no. 2-3, 2013, pp. 113–158.

[5] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele University, vol. 33, no. 2004, 2004, pp. 1–26.

[6] R. C. Motta, K. M. de Oliveira, and G. H. Travassos, "Characterizing interoperability in context-aware software systems," in 2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC). IEEE, 2016, pp. 203–208.

[7] S. Matalonga, F. Rodrigues, and G. H. Travassos, "Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review," Journal of Systems and Software, vol. 131, 2017, pp. 1–21.

[8] M. Petticrew and H. Roberts, Systematic reviews in the social sciences: A practical guide. John Wiley & Sons, 2008.

[9] S. Jiang, H. Zhang, C. Gao, D. Shao, and G. Rong, "Process simulation for software engineering education," in Proceedings of the 2015 International Conference on Software and System Process. ACM, 2015, pp. 147–156.

[10] G. Sindre, E.-A. Karlsson, and T. Stålhane, "Software reuse in an educational perspective," in SEI Conference on Software Engineering Education. Springer, 1992, pp. 99–114.

[11] F. Agalbato, "Design and implementation of robo3: an applied game for teaching introductory programming," Scuola di Ingegneria Industriale e dell'Informazione, 2017.

[12] T. Jordine, Y. Liang, and E. Ihler, "A mobile-device based serious gaming approach for teaching and learning java programming," in 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. IEEE, 2014, pp. 1–5.

[13] R. Atal and A. Sureka, "Anukarna: A software engineering simulation game for teaching practical decision making in peer code review." in QuASoQ/WAWSE/CMCE@ APSEC, 2015, pp. 63–70.

[14] Á. Serrano-Laguna, J. Torrente, B. M. Iglesias, and B. Fernández-Manjón, "Building a scalable game engine to teach computer science languages," IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 10, no. 4, 2015, pp. 253–261.

[15] E. V. Duarte and J. L. Pearce, "A cross-cultural review of lightbot for introducing functions and code reuse," Journal of Computing Sciences in Colleges, vol. 33, no. 2, 2017, pp. 100–105.

[16] R. Law, "Introducing novice programmers to functions and recursion using computer games," in European Conference on Games Based Learning. Academic Conferences International Limited, 2018, pp. 325–334.

[17] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with scratch," Computers & Education, vol. 120, 2018, pp. 64–74.

[18] N. Pellas and S. Vosinakis, "Learning to think and practice computationally via a 3d simulation game," in Interactive Mobile Communication, Technologies and Learning. Springer, 2017, pp. 550–562.

[19] M. Aedo Lopez, E. Vidal Duarte, E. Castro Gutierrez, and A. Paz Valderrama, "Teaching abstraction, function and reuse in the first class of cs1: A lightbot experience," in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education. ACM, 2016, pp. 256–257.

[20] N. Boldbaatar and E. Şendurur, "Developing educational 3d games with starlogo: The role of backwards fading in the transfer of programming experience¡? aq1?¿," Journal of Educational Computing Research, vol. 57, no. 6, 2019, pp. 1468–1494.

[21] N. Pellas, "The development of a virtual learning platform for teaching concurrent programming languages in the secondary education: The use of open sim and scratch4os," Journal of e-Learning and Knowledge Society, vol. 10, no. 1, 2014, pp. 129–143.

[22] A. Sierra, T. Ariza, F. Fernández-Jiménez, J. Muñoz-Calle, A. Molina, and Á. Martín-Rodríguez, "Educational resource based on games for the reinforcement of engineering learning programming in mobile devices," in 2016 Technologies Applied to Electronics Teaching (TAEE). IEEE, 2016, pp. 1–6.

[23] E. Lee, V. Shan, B. Beth, and C. Lin, "A structured approach to teaching recursion using cargo-bot," in Proceedings of the tenth annual conference on International computing education research. ACM, 2014, pp. 59–66.

[24] C.-K. Chang, "Effects of using alice and scratch in an introductory programming course for corrective instruction," Journal of Educational Computing Research, vol. 51, no. 2, 2014, pp. 185–204.

[25] D. Bau, "Droplet, a blocks-based editor for text code," Journal of Computing Sciences in Colleges, vol. 30, no. 6, 2015, pp. 138–144.

[26] D. A. Bau, "Integrating droplet into applab—improving the usability of a blocks-based text editor," in 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond). IEEE, 2015, pp. 55–57.

[27] H. Chandrashekar, A. G. Kiran, B. Uma, and P. Sunita, "Introducing programming using "scratch" and "greenfoot"," Journal of Engineering Education Transformations, 2018.

[28] S. Yamashita, M. Tsunoda, and T. Yokogawa, "Visual programming language for model checkers based on google blockly," in International Conference on Product-Focused Software Process Improvement. Springer, 2017, pp. 597–601.

[29] J. Vincur, M. Konopka, J. Tvarozek, M. Hoang, and P. Navrat, "Cubely: Virtual reality block-based programming environment," in Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, ser. VRST '17. New York, NY, USA: ACM, 2017, pp. 84:1–84:2. [Online]. Available: http://doi.acm.org/10.1145/3139131.3141785

[30] L. A. Vaca-Cárdenas, F. Bertacchini, A. Tavernise, L. Gabriele, A. Valenti, D. E. Olmedo, P. Pantano, and E. Bilotta, "Coding with scratch: The design of an educational setting for elementary pre-service teachers," in 2015 International Conference on Interactive Collaborative Learning (ICL). IEEE, 2015, pp. 1171–1177.