J. Figueiredo and F. J. García-Peñalvo, "Teaching and learning strategies of programming for university courses," in TEEM'19 Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality (Leon, Spain, October 16th-18th, 2019), M. Á. Conde-González, F. J. Rodríguez-Sedano, C. Fernández-Llamas and F. J. García-Peñalvo, Eds. ICPS: ACM International Conference Proceedings Series, pp. 1020-1027, New York, NY, USA: ACM, 2019. doi: 10.1145/3362789.3362926.

Teaching and learning strategies of programming for university courses

José Figueiredo
Research Unit for Inland Development
Polytechnic of Guarda
Portugal
jfig@ipg.pt

Francisco José García-Peñalvo
Computer Science Department Research Institute for Educational Sciences GRIAL research group
University of Salamanca
Spain
fgarcia@usal.es

ABSTRACT

It is consensual to consider teaching and learning programming difficult. A lot of work, dedication, and motivation are required for teachers and students. Since the first programming languages have emerged, the problem of teaching and learning programming is studied and investigated. The theme is very serious, not only for the important concepts underlying the course but also for the lack of motivation, failure, and abandonment that such frustration may imply in the student. Immediate response and constant monitoring of students' activities and problems are important. With this work, it is our goal to improve student achievement in courses where programming is essential. We want each student to be able to improve and deepen their programming skills, performing a set of exercises appropriate and worked for each student and situation. We intend to build a dynamic learning model of constant evaluation, build the profile of the student. The student profile will be analyzed by our predictive model, which in case of prediction of failure, the student will have more careful attention. Predict the student's failure with anticipation and act with specific activities, giving the student the possibility of training and practicing the activities with difficulties. With this model, we try to improve the skills of each student in programming.

KEYWORDS

programming, teaching programming, learning programming, CS0, CS1, datasets, neural networks.

1 Context and motivation

Every year, over 25 years of teaching introduction to programming, we feel the same problems and difficulties of the students. Every year we make small adaptations and experiences in the way of teaching, according to the students' previous skills and knowledge. However, almost invariably, the result is the same, high rates of failure, lack of interest, and consequent dropout.

The initial learning of programming is one of the great obstacles that students face when they start an Engineering course. In addition to the inherent difficulties at the beginning of a new cycle of studies, problems in the initial learning of programming are one of the main causes of discouragement and the failure of students. Several factors may be at the origin of this problem, such as the lack of capacity for abstraction, the lack of preparation for the mental construction of the reason necessary for the resolution of problems or the use of inadequate teaching and learning methods [3].

Our main motivation for the development of this work is to understand what difficulties students have, which factors most influence their learning programming process, which tools and/or methods or technologies can be used to reduce problems in the teaching and learning of the initial programming course. Develop a new learning environment of programming to help students to overcome their difficulties.

This work is an attempt to demonstrate the importance to recognize in the first beginning of learning programming the difficulties that student may have and analyze the effectiveness of the methods we propose to implement. On the other hand, we want our experience to contribute significantly to the entire Computer Science Education community.

The remainder of the paper is structured as follows. The next section describes the state of the art. Section 3 presents working hypothesis. Section 4 identify the principal objectives sought. Section 5 details the research approach and methods. Section 6 presents and discusses the results to date and their validity. Section 7 describes the dissertation status. And, finally, the current and expected contributions.

2 State of the art

The teaching and learning of introductory programming courses is a great challenge for all those who have to deal with this area. Programming is a process of transforming a mental plan of current terms into terms compatible with the computer [17]. When teaching computer programming, the main objective is to empower students with the skills needed to create computer programs that can solve real-world problems. In this context, programming requires quite particular characteristics and skills that students may struggle to obtain, often in a short period.

Among these, Jenkins [19] identified the following: the abstract concepts inherent to programming; the competencies and mental abilities required to decompose and solve problems; the use of specific syntax that students are required to memorize; and the semantics and structure new non-natural languages. As such, among the topics CS students are required to learn, computer programming is particularly difficult. This is a well-known fact, among the Computer Science Education (CSE) community [3]. Not only teaching computer programming is a recent area (when compared to other knowledge areas) but it is also a fast-changing one. Knowledge and tools rapidly become obsolete, making it harder to teach and learn even the basics. Since the emergence of the first programming languages that this phenomenon is studied. There are numerous studies carried out with the main reflections on the difficulties of solving programming problems [11, 19, 30–32]. The teaching of programming requires a methodology different from that of other disciplines. Theories such as active learning, learning by doing, peer assisted learning, or peer instruction [28], with good results in the most diverse teaching areas, also have confirmed results in this area [20, 27]. Other methodologies and techniques can be combined and applied in different environments and situations and articulated according to the needs and interests of the students. In the paper presented in [35], the authors present a systematic review of the articles that describe teaching approaches of introduction to programming, and those that provide a positive effect on the approval rates in these courses. From this work, according to their authors, are the best teaching practices, which on average can improve the success rates in introductory programming courses, compared with the traditional teaching of programming. In recent years, there has been a growing interest in teaching concepts of computer science to young people especially in the area of computational thinking. This interest is not only due to training for future work or high demand in this field but also because of the benefits that computer programming can bring to everyone. Computer programming and robotics, for example, play an important role in developing skills such as problem-solving, computational thinking, and creativity. Computational thinking is a fundamental skill for everyone. The term computational thinking was made popular by Jeannette M. Wing [36]. The author defends the mass diffusion of computational thinking, just like reading, writing, and arithmetic. Computational Thinking is an aptitude that allows us to create solutions to problems by making use of computer techniques [13]. Computers can be used to solve problems. However, before a problem can be tackled, the problem itself and the ways in which it could be solved need to be understood. Computational thinking allows us to take a complex problem, understand what it is and develop possible solutions. There are innumerable initiatives of various entities and organizations with the aim of promoting the study of computational thinking and, consequently, programming [2, 5, 12, 14–16, 23, 24]. Perhaps in the near future, most students have gone through these initiatives of dissemination of computational thinking and programming. Consequently, some of the skills referred to as essential for learning programming have been more developed.

Recently, work has been done to predict student performance in the introduction to programming. These methods based on machine-learning methods, as described in [22], support vector machines are used to predict students' final exam scores based on data collected automatically for instructors. In [22] a prediction model, PreSS (Predict Student Success), based on machine learning algorithms, is proposed to predict students' success in introductory programming courses. In addition, the study analyzes and compares the performance of various machine learning methods.

3 Problem Statment

After a brief description, contextualization, and motivation on the subject of teaching difficulties and the initial learning of the programming, it is important to emphasize that the main motivation, in the accomplishment of this work, is to be able to improve our teaching quality. And, consequently, reduce the high rate of failure, abandonment, and lack of motivation of students in courses where programming is a fundamental component.

This research involved the students, from an introductory programming course taught in C, of Computer Science from the Polytechnic of Guarda, Portugal. The Polytechnic of Guarda (IPG) is an institution of higher education located in the interior of the country. Our study group has very special characteristics which may affect, in our opinion, the learning programming process:

- The course of computer engineering, IPG, is usually not the first choice of students, which in some circumstances affect students' motivation and engagement.
- Average grade, in recent years, is between 10 and 12 values.

- Students reveal some general difficulties in the area of CS.
- Many of our students have never had programming courses, nor the opportunity to practice computational thinking activities.

The main objectives to achieve with this work are:

- Determine the factors that most influence the teaching and learning process of initial programming.
- Identify the techniques or set of techniques used by teachers that most influence successful initial programming learning.
- Building the profile of the student in programming, according to the evaluation along the course of a set of variables.
- Define and evaluate a machine-learning predictive model of student failure based on the student profile.
- Create immediate action plans based on the classification of our predictive model.

4 Research Objectives

Based on the defined objectives, we formulated some research questions for which our study intends to answer.

- What kind of activities from computational thinking contribute positively to success in the introduction to programming?
- Different presentation for the same activity according to the student learning styles positively influence the success in programming learning?
- What are the factors that most influence the learning process of initial programming?
- What are the factors that best characterize the student's profile in programming?
- Is the predictive model of machine learning of student failure based on student profile efficient?

5 Research approach and methods

One of the main objectives of this study is to understand the factors that most influence success, or failure, in the teaching and learning of initial programming. This study will allow determining what are the methods, techniques, attitudes, and behaviors that students and teachers, particularly in the field of initial programming, with characteristics similar to those of our study model, improve the educational process. The need to investigate in education [22] arises when we want to know better the functioning of a certain educational situation and, we intend to answer multiple questions that we put on how to improve our way of acting. Specifically and according to [22], we must do research in education in the realization of the following actions:

- Respond to the need to know and improve a certain educational reality;
- Use new methods in teaching and analyze the effectiveness of the application of these methods, in order to improve an educational reality;
- Evaluate the situation studied and analyze the causes that led to a certain diagnosis;
- Generalize conclusions that may affect other individuals.

With the objective of improving the educational process, the methodology we will use is action research. According to [1, 21, 26] when a teacher engages in an investigation in the classroom, he acquires the action research designation. Second, any type of research involves asking questions, seeking valid and objective answers, interpreting and analyzing the results and applying those results to improve the educational process. In the process of action research, we identify three main phases: identify the problem under study; collect valid information; analyze and interpret the results with the aim of improving the process. The use of this methodology aims to improve learning practices that allow us to make a significant change. We can see this process of action research as a dynamic process in constant observation and evaluation [1, 21, 26]. This type of methodology can also be seen as a spiraling research process, where the definition of the problem and context, the planning of the action, the action and observation, the criticism and reflection result in a new redefinition of the problem and application of the following phases of the process to solve/improve the problem.

In the following subsections, they describe the work done in an attempt to respond to our problems.

6 Improving Computational Thinking

As noted, most of our students never had the opportunity to develop their computational thinking. Therefore, according to the known characteristics of the student, we will propose exercises that allow to develop their cognitive capacities of reasoning and spatial visualization, strongly associated with the characteristics necessary for the programming. Some of these activities referred to in the works we developed in [8–10, 29], such as: follow and give instructions, drawing maps, description maps and directions, origami, punched holes, and others. On the other hand, in order to answer one of the research questions formulated "What kind of activities from computational thinking contribute positively to success in the introduction to programming?", we developed an experiment to evaluate the results and the behavior of its students with this type of activities.

According to characteristics of the IPG computer engineering students, we have created a free course of pre-programming for improving positive effect on approval success rates in learning programming. This course is designed to provide students with a set of computational thinking exercises to substantially improve their cognitive abilities. The course is not mandatory and will function with teacher recommendation. The course session planning activity is:

- Follow and Give instruction.
- Map Design.
- Paper Folding and Origami.

6.1.1. *Follow and Give instruction.* The use of this kind of exercises has as purpose to increase the development of

students' cognitive reasoning abilities and spatial visualization, strongly associated with the characteristics necessary for programming [11, 33, 34].

Based on this methodology, which are also used to evaluate the ability of students to programming, we have developed exercises to work with students. Some examples:
Example number 1: Students should design on a paper what a student or a teacher describes.

1. On a sheet of paper draw a square measuring approximately 5 cm. on its sides.
2. Draw a small dot in the center of the square.
3. Draw a line that starts at the top right corner to the bottom left corner, passed by the point.
4. Draw a line that starts in the upper left corner to the bottom right corner, passing the point.
5. Write your first name in the triangle below the center point.

Example number 2: It is also possible to practice from an image, see Figure 1, asking students to describe it through the design of others images.

Figure 1: Examples for follow and give instruction.

6.1.2. *Map Design.* With the use of this type of exercises we aim to develop students' capacities in planning, designing and describe in terms of specific characteristics in a concrete situation. Studies have demonstrated the relationship between the style and the level of detail in the description and construction of a map with the objectives of a programming course [11]. These activities include exercises for the student to move from point A to point B, within our school for example. This type of activity also includes the design and / or the representation of a path in a map. In this exercise we will evaluate the level of detail and clarity in the resolution (see Figure 1).

6.1.3. *Paper Folding and Origami.* Origami and / or paper folding [6, 7, 18] is a Japanese secular art widespread throughout the world, known for the development of features, such as: visual and spatial perception, fine motor coordination, memory, relieving stress and tension, patience and persistence; self-confidence, logical thinking and attention and concentration. Some examples are presented in Figure 2.

7 Building the student profile

In order to find a solution to our problem, improve our way of acting in the teaching programming in university courses, increase students' motivation levels, reduce failure and contribute to the development of this area, and respond to the formalized research questions "What are the factors that most influence the learning process of initial programming?" and "What are the factors that best characterize the student's profile in programming?". We develop a prototype that includes the best proposals of success in the teaching and learning of the initial programming and that satisfies our particular needs. In **Figure 3**, we can see the draft of our proposal. Where we emphasize the building skills of each student. The construction of each student's competencies profile in programming is based on the concept of current video games such as FIFA 19 or Assassin's Creed, for example. The characters are invited to build and improve their characteristics and skills in specific areas to complete their tasks or change their level. For example, a FIFA player may train a penalty, dribble, free kicks and corner kicks practice and other actions to improve his or her abilities during the game. Likewise, we want each student to be able to improve and deepen their programming skills by performing a set of appropriate and worked exercises for each student and situation. It is our intention to be able to identify the minimum set of skills necessary for the success of the student in the course.

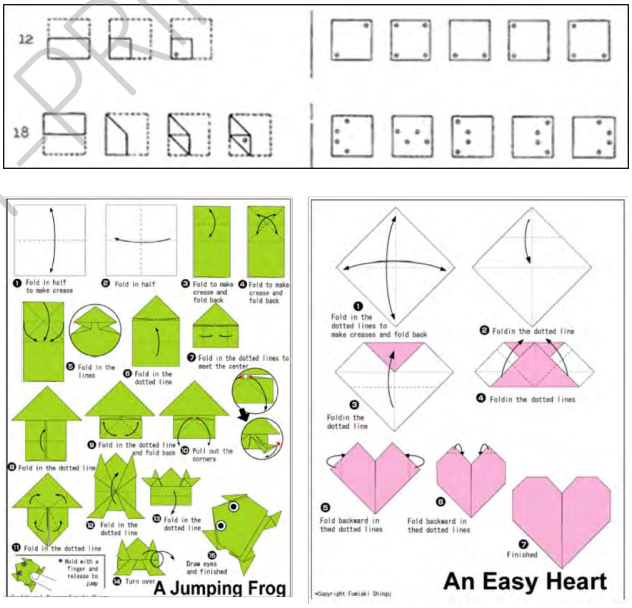

Figure 2: Examples of Origami and Punched Holes (adapted from [19]).

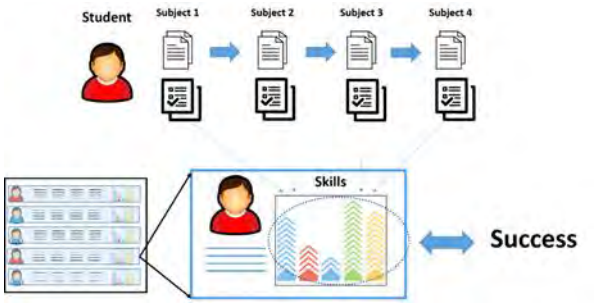

Figure 3: Representative scheme of our prototype - building the student profile in programming.

To build the skills of each student, we want to use diverse types of challenges with immediate feedback and consequent updating of their set of skills.

7.1.1. *Challenges.* Is important to know well the student and their background competences. In order to characterize the student, we will carry out a set of questionnaires related to some of the personal data, such as his age, his area of pre-university studies, his knowledge of programming and general knowledge in computer science. As in the work developed by [4], we consider it important to know each student's Index of Learning Styles. This way it will be possible to adapt some exercises to the student's learning style. The following group of challenges is directly related to the contents of the introductory programming course. In the various stages of initial programming learning, various types of challenges will be suggested and repeated according to each student's needs and level of requirement. Some examples of this type of challenges are Parson problems, gapped text, multiple matching, multiple choice, and others as mention in [25]. In addition, exercises that are more traditional like "Write a program that ..." with output perfectly identified and the possibility of test with different inputs. All exercises should have immediate feedback. According to the results, other types of exercises should be suggested, whether it is to improve a certain competence or to acquire new skills.

8 Predictive Model

In this section, we described a machine-learning predictive model of student failure based on the student profile, which is built throughout programming classes by continuously monitoring and evaluating student activities, described in the last subsection.

Table 1 describes the variables (attributes) collected in order to build each student profile. The data was then divided into training and test datasets. The training dataset was then used to build the model while the test dataset was used to validate it. The resulting model allows teachers to early identify students that are more likely to fail, allowing them to devote more time to those students and try novel strategies to improve their programming skills. The proposed predictive model of student failure based on the student profile is represented in a schematic form in Figure 4.

9 Results to date and their validity

The main purpose of this section is to justify and validate our claims.

10 Results for Improving Computational Thinking

Two online questionnaires were developed. One with 20 questions about Punched Holes (PH) and another with 5 Follow and Give Instruction (FGI) activities. The number of students who answered the questionnaire was 46. The PH questionnaire was evaluated by the number of correct answers. The FGI of activities were classified according to the following evaluation scale from not responding (value 0) to detailed description and using references (value 4). The results of Punched Holes (PH) activity are shown in the table \ref{tab1}. In the first line, PH < 10, 9 of the 46 students had a score of less than 10 correct answers. Of the 9 students, 6 successfully completed the course, which corresponds to a rate of 66.7%. For a score greater than 10 correct answers corresponds to a rate of 67.6%, and for a score greater an equal than 14 correct answers correspond to a rate of 77.8%. As it can easily be seen, in Table 2, we highlight the results for FGI >= 3, 20 students. Having successfully completed the course 18 students, which corresponds to 90%. InTable **3**, we combine the results of the two activities. It is important to enhance the results combined with the FGI activity (see Table **4**), of values greater than or equal to 3. The results are 100%, 88.9% and 100%, for the different PH results. These results lead us to believe that Follow and Give Instruction computational thinking activity has a strong influence on the success of the course. This experiment and results are described in [9].

Table 1. Student profile attributes collected using information provided by the students and the teacher.

Attribute	Description
participation	Participation in classes
curiosity	Curiosity and initiative in doing new activities
perfectionist	Try to write programs in the best possible way
passion	The passion, joy, and admiration, demonstrated by the student, towards programming
spatial_ability	Score in activities related to the detection of cognitive reasoning abilities and spatial visualization (*e.g.* punched holes)
follow_instructions	Capacity of give and following instructions
basics	Capacity to define the structure of a program, identifiers, data type, read and write data
if	Capacity of using an `if` conditional structure
switch	Capacity of using a `switch` structure
for	Capacity of using the a `for` iterative structure
do-while	Capacity of using a `do-while` structure
while	Capacity of using the a `while` structure
activity_01	Activities to evaluate the students knowledge using different C programming structures
activity_02	
activity_03	
activity_04	
activity_05	

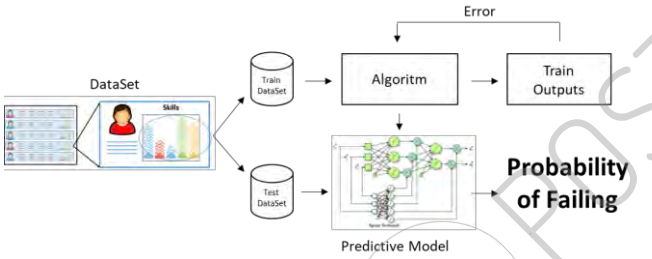

Figure 4: Predictive model of student failure based on the student profile.

Table 2. Results PH questionnaire.

		Success	%
PH < 10	9	6	66.7%
PH >= 10	37	25	67.6%
PH >= 14	27	21	77.8%

Table 3. Results Follow and Give Instructions (FGI) activities.

		Success	%
FGI < 3	26	13	50.0%
FGI >= 3	20	18	90.0%

Table 4. Results for PH and FGI.

		Success	%
PH < 10 and FGI < 3	7	4	57.1%
PH < 10 and FGI >= 3	2	2	100.0%
PH >= 10 and FGI < 3	19	9	47.4%
PH >= 10 and FGI >= 3	18	16	88.9%
PH >= 14 and FGI < 3	10	6	60.0%
PH >= 14 and FGI >= 3	15	15	100.0%

11 Results for Predictive Model

For evaluating the performance of the classifier model and asserting its quality, we use the accuracy and precision metrics, expressed as percentages. These metrics are based on a confusion matrix, see Table 5, containing the number of correctly and incorrectly classified examples for each class, in the form of true positives (TP) - that represents the students who failed and the classifier confirms, true negatives (TN) - which means that the students succeeded and the classifier confirms, false positives (FP) - in reality the students have succeeded and are classified as fail, and false negatives (FN) - students were classified as having success but failed. A perfect classifier should only present non-zero values in the confusion matrix main diagonal, as these correspond to correct classifications, while the remaining "cell" values represent mis-classified samples.

Table 5. Confusion matrix for binary classification.

		Observed	
		Student Fail	Student Success
Classified	Student Fail	TP	FP
	Student Success	FN	TN

In order to build a neural network predictive model for student failure, Multiple Back-Propagation software, available at http://mbp.sourceforge.net/ was used.

The application of our propose of machine learning predictive model of student failure, the results obtained are expressed as a percentage. These results are subject to a normalization, which transforms all values with a percentage greater than 50%, are classified as 1 (failure), the others as 0 (success). After normalization, we calculate the values according to the confusion matrix for binary classification, at the end we obtain the results present in Table 6. In our experiments, the most problematic value is the students who fail, and is classified as success (FN). According to the model, these students will not be given special attention because the model predicts they will succeed. However, in our model, this value is down 1.96% which corresponds to 1 student. The results show, that the number of correct decisions that the classifier makes, or the percentage of all students that are correctly classified, the accuracy is 94,1%, the precision is 95,5%, which represents the correctly classified students.

Table 6. Confusion matrix for binary classification, with the results obtained.

		Observed	
		Student Fail	Student Success
Classified	Student Fail	42 (82.35%)	2 (3.92%)
	Student Success	1 (1.96%)	6 (11.76%)

12 Dissertation Status

The first barrier for most engineering students is learning to program. As described earlier in this document, several factors may be at the root of this problem. Moreover, this phenomenon is universal. It is not specific to a course, school or country. Understanding students' difficulties and responding with methods and tools to help them overcome them is the main objective of this work. Although there are numerous research papers, tools and methods that improve teaching and learning, the results have not improved. We are convinced that this is due to the different characteristics of the students, where, consequently, they require different teaching and learning methods. An automatic tool that can detect students' difficulties will allow the student to get important feedback from their skills. And, it will provide the teacher with a quick means of detecting difficulties, so that he can act in an effective and personalized way.

So far:

- We identify the problem and find that it is a universal problem.
- We identify our goals and define the research questions.
- In an attempt to answer our research questions, we construct the student profile based on the set of activities developed throughout the course and construct a predictive model of machine learning of student failure based on the student profile.
- We believe the state of the art can be improved. It is necessary to analyze each work with activities carried out and results achieved in reducing the difficulties of students in the courses of initial programming.
- Our study group has small dimensions. These are the students of the computer course at the Polytechnic of Guarda. We need to extend the study group to more students.
- It is necessary to improve data collection. It is necessary to collect as much data as possible from the activities carried out by the students. And in an automatic way, only in this way will it be possible to achieve the goals proposed.
- It is necessary to construct the specific action plans according to the topics of the subject and the proficiency profile of the student.

13 Current and expected contributions

We find our work of great utility to find the solution to the problems of learning from initial programming. The dynamic and constant analysis of the students in programming, based on the construction of the profile of the student, will allow to evaluate the student at each moment of his course and to act in an immediate way. This model will allow early detection of students' difficulties and problems and encourage the student to improve and feel that they are being accompanied in a very personal way. Because the exercises are customized according to your needs and characteristics.

With the described model it will be possible to establish a relationship through the student profile in programming and the final results of the course of initial programming. A trained neural network that gives us the probability that a student with a certain profile will fail the course of initial programming, will be a precious tool to achieve the proposed goals.

This work will allow us to reflect on the importance of computational thinking in the student's competences, acquired over the years of student. And your influence on your future life as a student in courses where programming is essential. This need is justification for the need for the pre-programming course or clubs of computational thinking activities. For, as already described, many of the students who enter the course of initial programming, on the IPG, never had the opportunity to practice such computational thinking skills.

ACKNOWLEDGMENTS

This work is part of the PhD program in Informatics Engineering at the University of Salamanca, with the provisional title of "Learning Strategies and Learning Programming in University Students". Having as director Professor Francisco José García-Peñalvo.

REFERENCES

[1] Arends, R. 2005. *Learning to Teach*. McGraw-Hill Education.

[2] Basogain, X. et al. 2017. Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior*. (May 2017). DOI:https://doi.org/10.1016/j.chb.2017.04.058.

[3] Bergin, S. and Reilly, R. 2005. Programming: Factors that Influence Success. *SIGCSE '05: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, United States, 2005), 411–415.

[4] Carmo, L. et al. Learning styles and problem solving strategies.

[5] Cole, E. 2015. On Pre-requisite Skills for Universal Computational Thinking Education. (2015), 253–254. DOI:https://doi.org/10.1145/2787622.2787737.

[6] Cooper, S. et al. 2015. Spatial Skills Training in Introductory Computing. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. (2015), 13–20. DOI:https://doi.org/10.1145/2787622.2787728.

[7] Falomir, Z. 2016. Towards A Qualitative Descriptor for Paper Folding Reasonin. *Proc. of the 29th International Workshop on Qualitative Reasoning (QR'16)* (New York, USA, 2016).

[8] Figueiredo, J. et al. 2016. Ne-course for learning programming. *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM '16* (New York, New York, USA, 2016), 549–553.

[9] Figueiredo, J. and Garcia-Peñalvo, F.J. 2017. Desenvolver o Pensamento Computacional Usando Seguir e Dar Instruções. *TICAI 2017 TICs para el Aprendizaje de la Ingeniería*. O. da S. Alfonso Lago Ferreiro, André Fidalgo, ed. 101–108.

[10] Figueiredo, J. and García-Peñalvo, F.J. 2017. Improving Computational Thinking Using Follow and Give Instructions. *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM 2017* (New York, New York, USA, 2017), 1–7.

[11] Fincher, S. et al. 2005. Computer Science at Kent programming courses. 1 (2005).
[12] García-Peñalvo, F.J. et al. 2016. Evaluation Of Existing Resources (Study/Analysis). (Jan. 2016). DOI:https://doi.org/10.5281/ZENODO.163112.
[13] García-Peñalvo, F.J. 2016. What Computational Thinking Is. *Journal of Information Technology Research*. 9, 93 (2016).
[14] García-Peñalvo, F.J. and Mendes, A.J. 2017. Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*. (Dec. 2017). DOI:https://doi.org/10.1016/j.chb.2017.12.005.
[15] González-González, C.S. 2019. State of the art in the teaching of computational thinking and programming in childhood education. *Education in the Knowledge Society 20*.
[16] Grover, S. and Pea, R. 2013. Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*. 42, 1 (2013). DOI:https://doi.org/10.3102/0013189X12463051.
[17] Hoc, J.-M. and Nguyen-Xuan, A. 1990. Language Semantics, Mental Models and Analogy. *J.-M. Hoc, T. R. G. Green, R. Samurçay, & D. J. Gilmore (Eds.), Psychology of Programming*. (1990), 139–156.
[18] Jaeger, A.J. et al. 2015. What Does the Punched Holes Task Measure? (2015).
[19] Jenkins, T. 2002. On the Difficulty of Learning to Program. *Language*. 4, (2002), 53–58. DOI:https://doi.org/10.1109/ISIT.2013.6620675.
[20] Kelly, J.O.' et al. An Overview of the Integration of Problem Based Learning into an existing Computer Science Programming Module.
[21] Kemmis, S. et al. 1982. *The Action Research Planner*. Deakin University.
[22] Liao, S.N. et al. 2019. A Robust Machine Learning Technique to Predict Low-performing Students. *ACM Transactions on Computing Education*. 19, 3 (2019), 1–19. DOI:https://doi.org/10.1145/3277569.
[23] Lye, S.Y. and Koh, J.H.L. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*. 41, (2014), 51–61. DOI:https://doi.org/10.1016/j.chb.2014.09.012.
[24] Mason, D. et al. 2016. Computational Thinking as a Liberal Study. *Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE '16)*. (2016), 24–29. DOI:https://doi.org/10.1145/2839509.2844655.
[25] Milková, E. 2015. Development of Programming Capabilities Inspired by Foreign Language Teaching. *Procedia - Social and Behavioral Sciences*. 171, (Jan. 2015), 172–177. DOI:https://doi.org/10.1016/j.sbspro.2015.01.104.
[26] Mills, G. 2007. Action Research: A Guide for the Teacher Researcher. (2007).
[27] Nuutila, E. et al. Learning Programming with the PBL Method - Experiences on PBL Cases and Tutoring.
[28] Porter, L. et al. 2014. Predicting student success using fine grain clicker data. *Proceedings of the tenth annual conference on International computing education research - ICER '14* (New York, New York, USA, 2014), 51–58.
[29] QUITÉRIO FIGUEIREDO, J.A. 2017. Cómo mejorar el pensamiento computacional: un estudio de caso. *Education in the Knowledge Society (EKS)*. 18, 4 (Dec. 2017), 35. DOI:https://doi.org/10.14201/eks20171843551.
[30] Rojas-Lopez, A. and Garcia-Penalvo, F.J. 2018. Learning Scenarios for the Subject Methodology of Programming From Evaluating the Computational Thinking of New Students. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*. 13, 1 (Feb. 2018), 30–36. DOI:https://doi.org/10.1109/RITA.2018.2809941.
[31] Rojas-López, A. and García-Peñalvo, F.J. 2019. Personalized Education for a Programming Course in Higher Education. *Innovative Trends in Flipped Teaching and Adaptive Learning*. M.L. Sein-Echaluce et al., eds. IGI Global. 203–227.
[32] Shuhidan, S. et al. 2009. A taxonomic study of novice programming summative assessment. *Conferences in Research and Practice in Information Technology Series*. 95, (2009), 147–156.
[33] Simon et al. 2006. Predictors of success in a first programming course. *Proceedings of the 8th Australian conference on Computing education - Volume 52*. (2006), 189–196. DOI:https://doi.org/10.1145/953051.801357.
[34] Study, N.E. 2012. An Overview of Tests of Cognitive Spatial Ability. *66th EDGD Mid-Year Conference Proceedings*. (2012), 6.
[35] Vihavainen, A. et al. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. *Proceedings of the tenth annual conference on International computing education research - ICER '14*. (2014), 19–26. DOI:https://doi.org/10.1145/2632320.2632349.
[36] Wing, J.M. 2006. Computational Thinking [Pensamiento computacional]. *Communications of the Association for Computing Machinery (ACM)*. 49, 3 (2006), 33–35.