J. Figueiredo and F. J. García-Peñalvo, "Teaching and Learning Tools for Introductory Programming in University Courses," in Proceedings of the 2021 International Symposium on Computers in Education (SIIE) (23-24 September 2021, Málaga, Spain), A. Balderas, A. J. Mendes and J. M. Dodero, Eds., USA: IEEE, 2021. doi: 10.1109/SIIE53363.2021.9583623.

Teaching and Learning Tools for Introductory Programming in University Courses

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

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

***Abstract*—Difficulties in teaching and learning introductory programming have been studied over the years. The students' difficulties lead to failure, lack of motivation, and abandonment of courses. The problem is more significant in computer courses, where learning programming is essential. Programming is difficult and requires a lot of work from teachers and students. Programming is a process of transforming a mental plan into a computer program. The main goal of teaching programming is for students to develop their skills to create computer programs that solve real problems. There are several factors that can be at the origin of the problem, such as the abstract concepts that programming implies; the skills needed to solve problems; the mental skills needed to decompose problems; many of the students never had the opportunity to practice computational thinking or programming; students must know the syntax, semantics, and structure of a new unnatural language in a short period of time. In this work, we present a set of strategies, included in an application, with the objective of helping teachers and students. Early identification of potential problems and prompt response is critical to preventing student failure and reducing dropout rates. This work also describes a predictive machine learning (neural network) model of student failure based on the student profile, which is built over the course of programming lessons by continuously monitoring and evaluating student activities.**

Keywords—introductory programming, teaching programming, learning programming, CS1, intelligent tutoring system, neural networks, predict success

I. CONTEXT AND MOTIVATION

Over the many years dedicated to the initial teaching of programming, the situations experienced are countless. Learning and constant challenges, frustrations, and the most diverse manifestations of difficulties on the part of students, are the daily lives of all those who have the privilege of teaching introductory programming.

Works on the problem of teaching and learning programming difficulties have been a constant since the appearance of the first programming languages. What are the difficulties, what are the main factors, what are the methodologies, what are the techniques or tools that most influence the teaching and learning of programming, are some of the most used themes in research and investigation works. All of them with a single objective: to improve the teaching-learning system. That is, make most students acquire the basic concepts and techniques of creating and implementing computer programs to solve problems. This is our main motivation for the development of our work.

II. SATE-OF-THE-ART

Teaching and learning introductory programming courses are a great challenge for everyone involved in the process. The difficulties of teaching and learning programming are studied since the appearance of the first programming languages [8, 9, 11, 27, 39]. Dedication and constant hard work are required for success, regardless of the methods and techniques used. Programming is a process of transforming a mental plane of current terms into computer-compatible terms [24, 38]. When teaching computer programming, the main objective is to equip students with the skills needed to create computer programs that can solve real-world problems. In this context, programming requires very particular characteristics and skills that students may find difficult to obtain, often in a short period.

A wide variety of themes have been explored in order to improve the teaching-learning process of programming. Some works suggest that the constant practice of computational thinking activities helps the development of useful skills for learning programming [22, 30].

The analysis and diagnosis of compiler error messages are one of the main topics addressed in recent years [1, 4–7, 33].

The use of automated assessment tools to provide feedback to students is another common theme. Where the main purpose of automated assessment is to compare computing results using a fixed set of test cases [2, 19, 40]. Educational practices of data mining and machine learning (AM) are concepts increasingly used in the computing field, in order to monitor the entire teaching and learning process and predict student success [3, 12, 21, 25, 28, 36].

The high failure rates in introductory programming courses are another of the main problems identified in the most diverse research works, like [29, 34, 37].

III. PROBLEM STATEMENT

A large percentage of students do not acquire basic skills in introduction to programming. This problem results in high failure rates, lack of motivation, and drop out of students in the introductory programming curricular unit of the computer science course.

On the other hand, due to many students, it is difficult for teachers to have a correct perception of the knowledge and difficulties of each student, and to intervene quickly with the personalized help needed by each student.

In a short period of time, students have to acquire a set of unusual skills. They must learn the syntax and semantics of a programming language and finally create and organize the different elements to solve problems. In this teaching and

learning process, many hours of dedication and work are required from students and teachers. Therefore, the existence of a technological tool to support the teaching and learning process of initial programming is necessary.

IV. RESEARCH OBJECTIVES

This review aims to explore the literature on teaching and learning introduction to programming, especially in higher education, through the identification of publications of interest to the computing community. Also, the contributions of these publications and the evidence of research results. We defined the following research questions:

RQ1. What teaching and learning problems of introductory programming in higher education have been the focus of literature?

RQ2. What evidence was reported in addressing different introductory programming problems?

RQ3. What methods are used to monitor the teaching and learning process of introductory programming?

V. RESEARCH APPROACH AND METHODS

After identifying the research problem and reviewing the literature, the next step is to make a preliminary choice of methodology. Considering some concepts about the perspective of quantitative research [28], such as it follows a positivistic epistemology which defends that there is an objective reality that can be expressed numerically, a representative sample allows the generalization of the results and, still, the quantitative perspective emphasizes studies that are experimental in nature, attaches importance to measures, and seeks relationships. About the research methodology, the type of action research recognized as a methodology that seeks to improve practices through change and learning from the consequences of these changes. It also allows the participation of all those involved. It develops in a spiral of cycles of planning, action, observation, and reflection. It is, therefore, a systematic learning process oriented towards action with the objective of reaching a certain end, requiring that it be submitted to the test, allowing to give a justification from the work, through a developed, proven, and scientifically examined argumentation.

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

A. Study group

This study involves a group of students from an introductory programming course to programming (Introduction to Programming), in the last 3 years, students of the Computer Science course at the Polytechnic of Guarda (IPG), Portugal. In this course, the C language is used to teach basic programming concepts. The number of students per year is around 105 students on average.

Our study group has very special characteristics. The computer course, IPG, is generally not the first choice of students; the average candidacy grade is between 10 and 12 points; and, in the last 3 years, we have received students from Portuguese-speaking African countries (Portuguese: Portuguese Speaking African Countries; PALOP), with several problems in their general education.

B. Data collection methods and tools

Data collection is performed through the HTProgramming application, built for this purpose. Data are collected, preferably in the classroom, through individual activities performed by the student. The set of results obtained by each student will be used to build their profile. At the same time, the dataset is used to train a neural network, which will be used to predict the failure of each student. In table I we can see the variables/attributes that define the student's profile and dataset used in the neural network.

TABLE I. STUDENT PROFILE ATTRIBUTES COLLECTED USING HTPROGRAMMING.

Attribute	Description
attendance	attendance to classes
student programming	student profile; previous course; computer and programming knowledge
spatial ability	score in activities related to the detection of cognitive reasoning abilities and spatial visualization
introductory concepts	score in activities related to introductory programming concepts, like data types e identifiers, identify errors
parson problems	score in activities related to parson problems (basic programs in c, data input/output, sequential instructions)
basic	score in activities related to building simple programs in c, data input/output, output formatted, sequential instructions
conditions	score in activities related to building programs in c with conditional structures
loops	score in activities related to building programs in c with loops
arrays	score in activities related to building programs in c with arrays manipulation
advanced	score in activities related to building programs in c for problem-solving

C. HTProgramming

HTProgramming - Help To Programming, is an application that aims to help students and teachers throughout the teaching and initial learning process of programming. In Figure 1 we present the general scheme of the HTProgramming application. The application was developed in Java language, with the Netbeans IDE. The desktop applicationa interacts with a remote MySQL database, in a Hosting Smart Linux service. The application consists of two modules: the administration module and the student module, with access to a remote database.

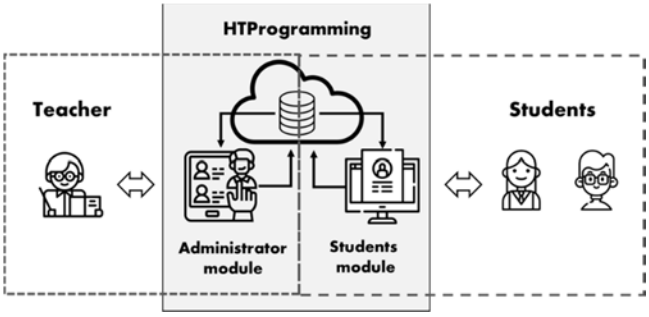

Fig. 1. General scheme of the application HTProgramming.

The administration module is used by the teacher to control the entire teaching and learning process. The teacher has access to individual student data and the activities performed. The teacher also has the possibility of inserting new activities or changing existing ones. The application

automatically generates the data to be used in the predictive system.

The student module is used by students taking the course. After identifying the student before the application, the student has at their disposal a set of activities directly related to the contents covered during the introduction to programming. In each activity performed, the student receives immediate feedback, scores obtained, suggestions for reading or reviewing, or even suggestions for new activities, and their profile is updated.

D. Activities performed

To answer our problem, we created a set of activities to be carried out by the student, to build their learning profile. As described above, these results are used to build the predictive model of student failure based on a Neural Network, described in more detail in [15]. In the following subsections, we briefly describe the activities performed.

1) Students characterization.

We start with the characterization of each student. We question some personal data, such as age or city of origin. Next, we want to know what your area of study in secondary education is and how you assess your knowledge of programming and informatics in general.

2) Paper folding

Paper folding, particularly punch holes, is often used to investigate spatial visualization skills [26], a skill directly associated with programming. In this type of exercise, students should imagine that they are folding and unfolding paper. The figures on the left, in Figure 2, represent a square piece of paper being folded, and the last of these figures has one or two small circles drawn to show where the paper has been punched. The figure on the right shows the location of the holes when the paper is unfolded.

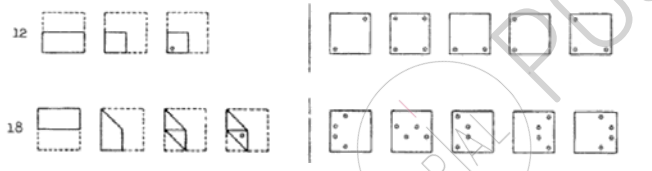

Fig. 2. Examples Punched Holes, adapted from REF.

3) Parson Problems

According to [10, 13, 31], one way to learn and practice an introduction to programming is to use Parson's Problems. Parson's problems are programming instructions in which the student must select, order, and indent code fragments. These tasks are great for the early stage of learning programming because students don't make syntax errors. In Figure 3, we can see an example of the Parson Problem.

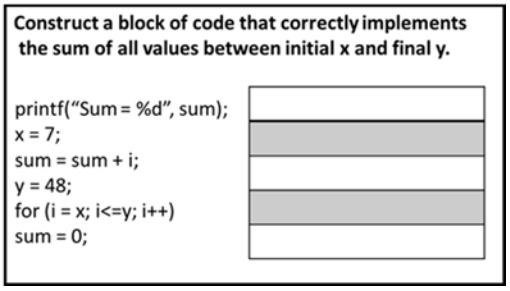

Fig. 3. Parson problem example.

4) Introductory concepts

Activities are related to introductory programming concepts, in the C language, such as data types, names and identifiers, and error identification. These types of activities are MCQ (Multiple Choice Question). In Figure 4 we can see an example.

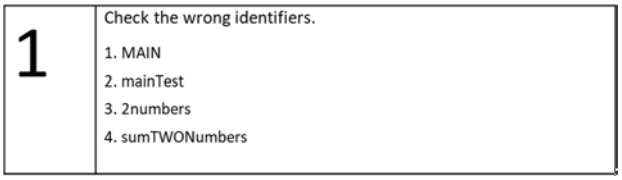

Fig. 4. Multiple Choice Question for introductory concepts example.

5) Coding Activities

In coding activities, students write code in C language, responding to suggested activities. Using the IDE (Integrated Development Environment) that they wish, they submit their proposed resolution in the application. The application runs the program with a set of verification tests, which results in a score. In Figure 5 we present an example of a basic activity and an advanced one in Figure 6, with the respective test cases. This set of activities is divided into:

a) Basic: activities related to building simple programs in c, data input/output, formatted output, sequential instructions.

b) Conditions: construction of programs with conditional structures.

c) Loops: building programs using for, while, do-while loops.

d) Arrays: activities related to building programs with matrix manipulation.

e) Advanced: problem solving using knowledge acquired in language c.

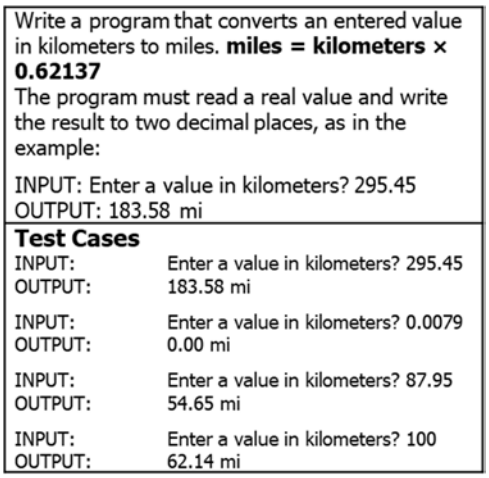

Fig. 5. Basic activity example, with test cases.

VI. RESULTS TO DATE

The HTProgramming application is the result of several years of experience and works with students. With this application, we are close to achieving a very useful tool for the initial teaching and learning process of programming.

To date, the results obtained are encouraging. However, the latest results obtained are compromised by the pandemic situation we are going through. The results obtained by the

HTProgramming application are well organized and in detail. The teacher has at his disposal a set of data that help him to monitor all student activity. For example, in Figure 7 the leader board, and in Figure 8 the individual results of each student.

Kaprekar constant, or 6174, is a constant that arises when we take a 4-digit integer, form the largest and smallest numbers from its digits, and then subtract these two numbers. Continuing with this process of forming and subtracting, we will always arrive at the number 6174.
Make a program that input a 4-digit integer, shows the different steps in arriving at the kaprekar constant.

Test Cases

INPUT:	Integer 4 digits? 1234
OUTPUT:	4231 - 1234 = 3087
	8730 - 0378 = 8325
	8532 – 2358 = 6174
INPUT:	Number (4 digits)? 4422
OUTPUT:	4422 - 2244 = 2178
	8721 - 1278 = 7443
	7443 - 3447 = 3996
	9963 - 3699 = 6264
	6642 - 2466 = 4176
	7641 - 1476 = 6174

Fig. 6. Advanced activity example, with test cases.

Table II shows the results of the last 3 academic years, highlighting the high number of students who passed the last academic year, 61 students, which corresponds to a percentage of 54.4%. This value is quite different, for the better, from the values of previous years. The main justification for this value was the online assessment motivated by the pandemic situation, which allowed a greater number of students to perform the assessment and less control in the students' tests. For the use of the predictive model, only students attending classes and recording activities in the HTProgramming application were considered, that is, 64 students. In relation to the total number of students enrolled, in the academic year 2020-2021, there is a percentage of 57.1%, of these students the number of approved students was 43, which is equivalent to a percentage of 38.4%.

This application includes a predictive Neural Network (NN) model of student failure based on student profiles collected during classes. For the last year, Figure 9 presents the resulting NN model confusion matrix in the test dataset. In addition, Table III presents the performance of the model on the test data.

Table IV presents the results of an analysis of the causes and effects of the different variables collected. We used the Correlation Coefficient of Pearson between each of the variables and the final results of learning. The interpretation of the correlation results [32] was based on the rules in table V. According to the results obtained, the coding activities are those that have a correlation coefficient classified as moderate or very high correlation.

VII. Dissertation Status

Our work is dependent on obtaining student data and the data collection tool. On the other hand, the curricular unit is an integral part of the curricular plan of the Computer Engineering course, in the 1st year and in the 1st semester, so if something does not happen as expected, we must wait until next year for the experience to be carried out.

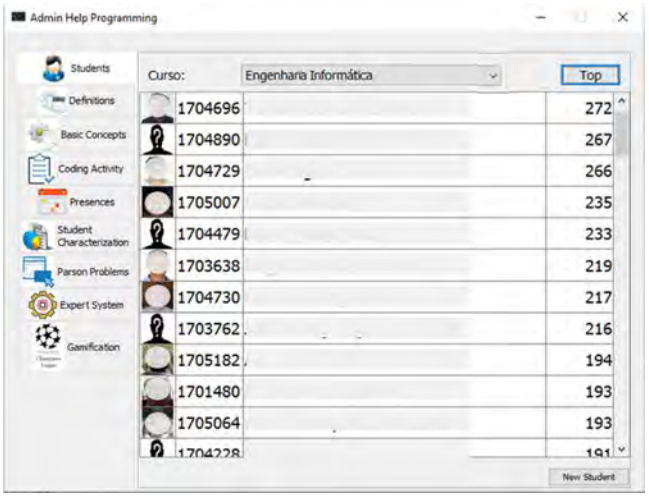

Fig. 7. Example of presentation of results by HTProgramming - leaderboard.

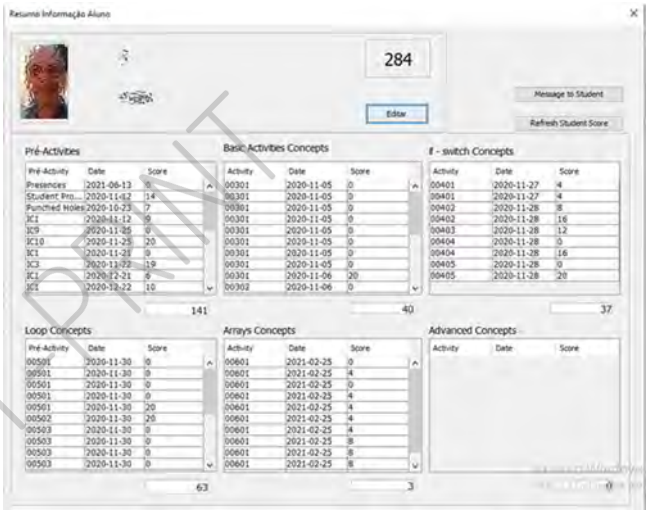

Fig. 8. Example of presentation of results by activity of each student.

TABLE II. Total Students per Academic Year and Number of Approved.

Academic year	Total students	Students approved
2018 - 2019	85	30 (35,3%)
2019 - 2020	119	20 (16,8%)
2020 - 2021	112	61 (54,4%)

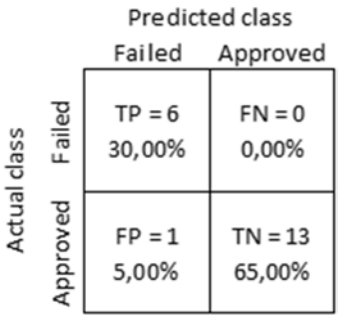

Fig. 9. Confusion matrix of the NN model for predicting student success in the dataset.

The results shown are the values obtained in the first use of the application. It is important to mention that the unthinkable outbreak of COVID-19 had a strong impact on

our lives and on the normal functioning of the teaching and learning process [20]. All the work developed was designed for a classroom and face-to-face environment. Somehow the results were compromised. However, the appearance of this pandemic, in which most of the context of the classes was in hybrid or blended teaching modality, led us to reflect [20, 23] and feel the need to adapt our teaching and learning model to this type of teaching.

TABLE III. PERFORMANCE OF THE NN PREDICTIVE MODEL FOR STUDENT FAILURE.

Metrics	Results
Accuracy	95,00%
Precision	85,71%
Recall	92,31%
F-score	92,86%

TABLE IV. CORRELATION COEFFICIENT OF PEARSON (VARIABLES VS FINAL RESULTS AND INTERPRETATION.

Variable	Correlation Coefficient of Pearson (Variable and Final Result)	Interpretation
attendance	0,19	Negligible correlation
student programming	0,31	Low positive correlation
spatial ability	0,11	Negligible correlation
introductory concepts	0,29	Negligible correlation
parson problems	0,49	Low positive correlation
basic	0,67	Moderate positive correlation
conditions	0,55	Moderate positive correlation
loops	0,56	Moderate positive correlation
arrays	0,93	Very high positive correlation
advanced	0,92	Very high positive correlation

TABLE V. RULES FOR INTERPRETING THE SIZE OF A CORRELATION COEFFICIENT.

Size of Correlation	Interpretation
0.9 to 1.0	Very high positive correlation
0.7 to 0.9	High positive correlation
0.5 to 0.7	Moderate positive correlation
0.3 to 0.5	Low positive correlation
0.0 to 0.3	Negligible correlation

In general, we believe that our model of teaching and learning introductory programming meets our goals. We need to review the set of coding activities and respective test cases. Technical improvements to the HTProgramming application prototype are also needed.

VIII. CURRENT AND EXPECTED CONTRIBUTIONS

We consider our work an important contribution to the problem of teaching and early learning in programming. The dynamic and constant analysis of students, based on the construction of the student's profile, will allow the student to be evaluated at every moment of their learning path and to act immediately. On the other hand, the proposed predictive Neural Network model of student failure will be a good method of early detection of students with greater difficulties.

Over the last few years, we have contributed to reporting and publishing our research work in the area. We started with some work in computational thinking, and the inclusion of activities before the introductory course on programming [14, 18, 35]. Then we started the study with the experience of constantly monitoring the student throughout the teaching and learning process [17], passing using gamification techniques to increase student motivation [16]. Finally, the use of automatic means for recording and assisting in the teaching and learning process of introductory programming.

ACKNOWLEDGMENT

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] Ahmed, U.Z. et al. 2018. Compilation error repair: For the student programs, from the student programs. Proceedings - International Conference on Software Engineering (May 2018), 78–87.

[2] Aldriye, H. et al. 2019. Automated Grading Systems for Programming Assignments: A Literature Review.

[3] Altabrawee, H. et al. 2019. Predicting Students' Performance Using Machine Learning Techniques. JOURNAL OF UNIVERSITY OF BABYLON for Pure and Applied Sciences. 27, 1 (Apr. 2019), 194–205. DOI:https://doi.org/10.29196/jubpas.v27i1.2108.

[4] Becker, B.A. 2016. A new metric to quantify repeated compiler errors for novice programmers. Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE (Jul. 2016), 296–301.

[5] Becker, B.A. et al. 2019. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE (Dec. 2019), 177–210.

[6] Becker, B.A. et al. 2018. Fix the first, ignore the rest: Dealing with multiple compiler error messages. SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education (Feb. 2018), 634–639.

[7] Becker, B.A. et al. 2018. The effects of enhanced compiler error messages on a syntax error debugging test. SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education (Feb. 2018), 640–645.

[8] Bennedsen, J. and Caspersen, M.E. 2019. Failure Rates in Introductory Programming: 12 Years Later. ACM Inroads. 10, 2 (2019), 30–36. DOI:https://doi.org/10.1145/3324888.

[9] Bennedsen, J. and Caspersen, M.E. 2007. Failure rates in introductory programming. ACM SIGCSE Bulletin. 39, 2 (Jun. 2007), 32–36. DOI:https://doi.org/10.1145/1272848.1272879.

[10] Denny, P. et al. 2008. Evaluating a new exam question: Parsons problems. Proceedings of the fourth international workshop on Computing education research. (2008), 113–124. DOI:https://doi.org/10.1145/1404520.1404532.

[11] Difficulties in learning programming: Views of students: 2012. https://www.researchgate.net/publication/267338258_Difficulties_in_learning_programming_Views_of_students. Accessed: 2019-05-16.

[12] Enughwure, A.A. and Ogbise, M.E. 2020. Application of Machine Learning Methods to Predict Student Performance: A Systematic Literature Review. International Research Journal of Engineering and Technology. (2020).

[13] Ericson, B.J. 2014. Adaptive Parsons Problems with Discourse Rules. Icer '14. (2014), 145–146. DOI:https://doi.org/10.1145/2632320.2632324.

[14] 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.

[15] Figueiredo, J. et al. 2019. Predicting Student Failure in an Introductory Programming Course with Multiple Back-Propagation. Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM'19 (New York, New York, USA, 2019), 44–49.

[16] Figueiredo, J. and García-peñalvo, F.J. 2020. Increasing student motivation in computer programming with gamification. 2020 IEEE Global Engineering Education Conference (EDUCON) (Porto, 2020), 997–1000.

[17] Figueiredo, J. and García-Peñalvo, F.J. 2018. Building Skills in Introductory Programming. Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM'18 (New York, New York, USA, 2018), 46–50.

[18] 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.

[19] Gao, J. et al. 2016. Automated feedback framework for introductory programming courses. Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE (Jul. 2016), 53–58.

[20] García-Peñalvo, F. et al. 2020. La evaluación online en la educación superior en tiempos de la COVID-19. Education in the Knowledge Society (EKS). 21, (2020). DOI:https://doi.org/10.14201/eks.23013.

[21] Gil, P.D. et al. 2020. A data-driven approach to predict first-year students' academic success in higher education institutions. Education and Information Technologies. (Mar. 2020). DOI:https://doi.org/10.1007/s10639-020-10346-6.

[22] 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.

[23] González, A.-B. et al. 2013. Experimental Evaluation of the Impact of B-Learning Methodologies on Engineering Students in Spain. Comput. Hum. Behav. 29, 2 (2013), 370–377. DOI:https://doi.org/10.1016/j.chb.2012.02.003.

[24] 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.

[25] Ihantola, P. et al. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. ITiCSE-WGP 2015 - Proceedings of the 2015 ITiCSE Conference on Working Group Reports (Jul. 2015), 41–63.

[26] Jaeger, A.J. et al. 2015. What Does the Punched Holes Task Measure? (2015).

[27] Jenkins, T. 2002. On the Difficulty of Learning to Program. Language. 4, (2002), 53–58. DOI:https://doi.org/10.1109/ISIT.2013.6620675.

[28] 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.

[29] Lindoo, E. 2018. Back to the Basics in an Effort to Improve Student Retention in Intro to Programming Classes. J. Comput. Sci. Coll. 34, 2 (2018), 72–79.

[30] Montes-León, H. et al. 2020. Mejora del Pensamiento Computacional en Estudiantes de Secundaria con Tareas Unplugged. Education in the Knowledge Society 21. (2020). DOI:https://doi.org/10.14201/eks.23002.

[31] Morrison, B.B. et al. 2016. Subgoals Help Students Solve Parsons Problems. Proceedings of the 47th ACM Technical Symposium on Computing Science Education. (2016), 42–47. DOI:https://doi.org/10.1145/2839509.2844617.

[32] Mukaka, M.M. 2012. Statistics Corner: A guide to appropriate use of Correlation coefficient in medical research.

[33] Prather, J. et al. 2017. On novices' interaction with compiler error messages: A human factors approach. ICER 2017 - Proceedings of the 2017 ACM Conference on International Computing Education Research (2017).

[34] Queirós, R. 2019. PROud-A gamification framework based on programming exercises usage data. Information (Switzerland). 10, 2 (2019), 1–14. DOI:https://doi.org/10.3390/info10020054.

[35] Quitério Figueiredo, J.A. 2017. How to Improve Computational Thinking: a Case Study. Education in the Knowledge Society. (2017).

[36] Rastrollo-Guerrero, J.L. et al. 2020. Analyzing and Predicting Students' Performance by Means of Machine Learning: A Review. Applied Sciences. 10, 3 (Feb. 2020), 1042. DOI:https://doi.org/10.3390/app10031042.

[37] 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.

[38] Sleeman, D. 1986. The challenges of teaching computer programming. Communications of the ACM. 29, 9 (Sep. 1986), 840–841. DOI:https://doi.org/10.1145/6592.214913.

[39] Watson, C. and Li, F.W.B. 2014. Failure Rates in Introductory Programming Revisited. Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (New York, NY, USA, 2014), 39–44.

[40] Zampirolli, F.A. et al. 2018. Evaluation process for an introductory programming course using blended learning in engineering education. Computer Applications in Engineering Education. 26, 6 (Nov. 2018), 2210–2222. DOI:https://doi.org/10.1002/cae.22029.