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

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 application 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.](image)

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.

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.

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.

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:

- **Basic**: activities related to building simple programs in c, data input/output, formatted output, sequential instructions.
- **Conditions**: construction of programs with conditional structures.
- **Loops**: building programs using for, while, do-while loops.
- **Arrays**: activities related to building programs with matrix manipulation.
- **Advanced**: problem solving using knowledge acquired in language c.

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.

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.

Table II. TOTAL STUDENTS PER ACADEMIC YEAR AND NUMBER OF APPROVED.

<table>
<thead>
<tr>
<th>Academic year</th>
<th>Total students</th>
<th>Students approved</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018 - 2019</td>
<td>85</td>
<td>30 (35.3%)</td>
</tr>
<tr>
<td>2019 - 2020</td>
<td>119</td>
<td>20 (16.8%)</td>
</tr>
<tr>
<td>2020 - 2021</td>
<td>112</td>
<td>61 (54.4%)</td>
</tr>
</tbody>
</table>

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

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

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.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>95.00%</td>
</tr>
<tr>
<td>Precision</td>
<td>85.71%</td>
</tr>
<tr>
<td>Recall</td>
<td>92.31%</td>
</tr>
<tr>
<td>F-score</td>
<td>92.86%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient of Pearson (Variable and Final Result)</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>attendance</td>
<td>0.19</td>
<td>Negligible correlation</td>
</tr>
<tr>
<td>student programming</td>
<td>0.31</td>
<td>Low positive correlation</td>
</tr>
<tr>
<td>spatial ability</td>
<td>0.11</td>
<td>Negligible correlation</td>
</tr>
<tr>
<td>introductory concepts</td>
<td>0.29</td>
<td>Negligible correlation</td>
</tr>
<tr>
<td>parson problems</td>
<td>0.49</td>
<td>Low positive correlation</td>
</tr>
<tr>
<td>basic</td>
<td>0.67</td>
<td>Moderate positive correlation</td>
</tr>
<tr>
<td>conditions</td>
<td>0.55</td>
<td>Moderate positive correlation</td>
</tr>
<tr>
<td>loops</td>
<td>0.56</td>
<td>Moderate positive correlation</td>
</tr>
<tr>
<td>arrays</td>
<td>0.93</td>
<td>Very high positive correlation</td>
</tr>
<tr>
<td>advanced</td>
<td>0.92</td>
<td>Very high positive correlation</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Size of Correlation</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9 to 1.0</td>
<td>Very high positive correlation</td>
</tr>
<tr>
<td>0.7 to 0.9</td>
<td>High positive correlation</td>
</tr>
<tr>
<td>0.5 to 0.7</td>
<td>Moderate positive correlation</td>
</tr>
<tr>
<td>0.3 to 0.5</td>
<td>Low positive correlation</td>
</tr>
<tr>
<td>0.0 to 0.3</td>
<td>Negligible correlation</td>
</tr>
</tbody>
</table>

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-Pejahlo.

REFERENCES


