Teaching of Programming - An Educational Performance Booster for Students at Economically Disadvantaged Schools

Paulo Brito[1], J. Antão B. Moura[2], Joaquim Honório[2], Marcelo Barros[2] and Igor Vieira[2]
[1]*Graduate Program in Computer Science, Federal University of Campina Grande (UFCG), Brazil*
[2]*Systems and Computing Department, Federal University of Campina Grande (UFCG), Brazil*

Keywords: Teaching of Programming, Educational Performance, Computer Science Education, Computational Thinking.

Abstract: The technological advances made in recent years bring with them the importance of introducing computer courses in the school context, and with this, a need for digital inclusion of students from economically disadvantaged schools. This paper reports on research to introduce programming in such school curricula and to evaluate possible benefits of such introduction for students' motivation towards learning in general. The research was based on the creation of a methodology for designing and offering programming courses at public and private schools and to verify possible correlations between students' performance and schools' economic scenarios they find themselves in. Preliminary results indicate that students who participated in the courses at schools with economic restrictions and inferior quality of IT infrastructure may still present better results in the courses. In addition, there is evidence of gains in their motivation towards learnig other subjects. The paper details results, analyzes causes for them and illustrates and explores implications for participating students at schools operating across the economic spectrum.

1 INTRODUCTION

With globalization and the technological advances that have occurred in recent years, it is becoming increasingly important to incorporate subjects in the curriculum that meet the technological needs of the present century (Pagani, 2015). Thus, the teaching of programming at the levels of basic education has grown worldwide due to its benefits to the learning process. The benefits include general pedagogical aspects, such as the development of logical reasoning, new ways of thinking about a problem and interdisciplinary learning, among others (Barbosa, 2017).

However, there are many difficulties to the teaching of programming in the educational structure in developing countries. Some of theses difficulties, like: insufficiente number of computers, lack of knowledge/skills of the teachers, not enough copies of sotware and difficulty to integrate in instruction not only also affect some developed countries as also the teaching of other subjects, creating obstacles for the implementation of a good infrastructure of Information and Communication Technology (ICT) (Pelgrum, 2001). Furthermore, these difficulties are aggravated, especially when it comes to education in less favored or "economically disadvantaged" schools (Duarte, 2007). Here an "economically disadvantaged school" is a school that is poor, i.e., it faces problems and difficulties due to lack of (enough) money and other economic resources - such as infrastructure in general, but IT in particular - and it is usually located in a low-income community. That is the case of public elementary, middle or high schools run by the government in Brazil. The economic disadvantage typically exists between a private and a public school; but it may also appear amongst public schools: one school may be more economically disadvantaged than another.

Economically disadvantaged schools are often the result of various factors and/or the interactions of these factors. In this work, the following indicators, when evaluated in want, are assumed to characterize a disadvantaged school: (i) number of functioning computers available to students; (ii) existence of computer classes; (iii) the quality of technological resources such as (old, little memory) computers and

Brito, P., Moura, J., Honório, J., Barros, M. and Vieira, I.
Teaching of Programming - An Educational Performance Booster for Students at Economically Disadvantaged Schools.
DOI: 10.5220/0006785101550164
In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018)*, pages 155-164
ISBN: 978-989-758-291-2

Internet access; (iv) quality of the school's overall infrastructure.

Previous research concentrated on investigating programming teaching strategies and indicated promising results (Barbosa, 2017) (Pina and Rubio, 2017). Grover, Pea and Cooper (2016) studied factors (e.g. prior technology experiences and student demographic information) that may affect results of programming courses. While studies like Papadakis, Kalogiannakis, Orfanakis and Zaranis (2017) pointed out some of the mainly barriers encounter by novices in these courses. However, the literature is short of works that assess the impact of economical resources of schools on the performance metrics of programming courses – be theses metrics specific to programing or generic in academic terms. Such an assessment is the object of this paper.

This paper discusses research results on the impact the teaching of programming as a curricular subject may have on the level of learning the subject but also as a performance driver for the overall academic motivation of students at economically disadvantaged schools. For that, two research questions (RQs) are of interest: one that relates students' performce on the introductory programming course to their school facilities; and, the other that explores benefits to students' academic behavior in general. To address these questions, albeit preliminarly, experiments have been designed based on the code.org platform (Kalelioglu, 2015) for the teaching of programming at private and public schools with varying degrees of economic disadvantages in Northeastern Brazil. This paper reports on the methodology for the design, application and evaluation of results of a course on introductory programming concepts.

Besides contributing preliminary evidence to answer the RQs, this paper may open new research opportunities on the topic of teaching of programming and its implications.

The remainder of this paper is structured as follows. Section 2 highlights basic concepts and reviews related work. In section 3, the methodology and the models used in the research are presented. Section 4 offers and analyses results of the considered introduction to programming course. Finally, section 5 presents' conclusions and explores future work opportunities on the topic.

2 RELATED WORK

Currently, most students from any school spend a large part of their free time playing computer games. Hence, integrating games into school subjects can increase student interest by providing opportunities for them to learn while having fun (Grivokostopoulou, Perikos and Hatzilygeroudis, 2016). Computer games can be used to teach almost all areas of computer science, and researchers point out that they could effectively provide the most interesting learning environments for knowledge acquisition and construction (Sung and Hwang, 2013).

Some works have investigated how to apply game concepts to programming teaching. Pina and Rubio (2017) have shown that the use of educational systems based on games brings a significant advantage to students, improving motivation and the general interest in learning. Ibáñez et al. (2014) also investigated the use of a strategy based on games for the teaching of programming basic concepts to undergraduate students. Barbosa (2017) states that, in programming logic classes for 10-year-olds, improvements were made in logical reasoning, where students understood algorithm concepts and variables by analogies with situations that they have already experienced, such as following a cake recipe. However, the teaching-learning process of programming is not a trivial task. Rodrigues (2002) highlights several problems that may affect the teaching and learning process of algorithms and programming, including: i) difficulty in making students develop logical reasoning when they are accustomed to memorize content; and, ii) lack of motivation of students generated by their lack of preparation and discouragement when they believe that the course presents difficult obstacles to be overcome. Grover, Pea, and Cooper (2016) studied factors related to students, such as past experiences and demographic data that influence learning outcomes.

However, although the above related papers discuss strategies, benefits and success factors of programming teaching, little has been discussed about the possible relationship between students' academic performance in general (and on introductory programming concepts in particular) and conditions and resources made available by their schools for a programming course.

This work extends previous research by analyzing the influence an introductory programming course may have on the overall academic performance of students at economically challenged schools.

Table 1: Characterization of participating schools.

School	Functioning Computers	Total Number of Students in Proposed Course	Proportion of Computers for Each Student	Regular Computer Classes (1 – Yes; 0 – No)	Quality of Technology Resources (Computer and Internet Access: 1- Very Bad to 5- Very Good)	Quality of School Infrastructure in General (1-Very Bad to 5- Very Good))	Overall Indicator (Ranking Index - RI)
Private School	40	11	3.63	1	5	5	10.04
School A	10	12	0.83	1	4	4	6.12
School B	8	16	0.50	0	5	4	6.12
School C	9	17	0.52	1	4	3	5.39
School D	14	18	0.77	1	3	4	5.32
School E	14	20	0.70	0	3	3	4.27
School F	14	21	0.66	0	3	3	4.24

3 METHODOLOGY

The objective of the research reported here is to analyze the impact of teaching a programming course as a code.org application. More specifically, the research aims to investigate how students from econmically disadvantaged participating schools behave when they come in contact with this new pedagogic resource.

3.1 School Characterization

Participating schools were ordered from the least to the most disadvantaged (please, see Table 1). A total of 7 schools were included in the study: one private school and 6 public schools (A to F).

The characterization was done using the criteria of Table 1, each of which received a score according to the associated degree of importance (I): 1 (very low importance), 2 (low importance), 3 (medium importance), 4 (high importance) and 5 (very high importance). A brief description of each criterion follows.

- **Number of functioning computers available to students:** the number of computers that are in perfect working order and are intended for student use. Although websites such as QEdu.org provide data on the number of computers in public schools in Brazil, the actual number of computers in operation does not match that informed. The amount of computers that were seen as available during the course was used instead. This characteristic is defined as being very important because the number of computers can directly impact the performance of programming classes.
- **Computer classes:** Earlier work, such as Grover's, Pea and Cooper (2016), pointed out that previous experiences students may have had with technology may have a direct effect on learning. Thus, as a result of what was seen during the course, this characteristic was defined as being of medium importance. Because it is a categorical variable, we assign scores (N) considering the following possibilities: 0 - the school does not provide computer classes and 1 the school provides computer classes.
- **Quality of technological resources:** The quality of resources refers to the condition of the internet access and computers. As with the number of computers, this feature may be a result of a lack of school resources and may have a direct impact on teaching programming. Therefore, this characteristic is assigned a great importance. The score for this criterium corresponds to the quality of technological resources: 1 (very bad quality), 2 (bad quality), 3 (medium quality), 4 (good quality) and 5 (very good quality).

- **Quality of School Infrastructure:** The quality of school infrastructure refers to the perceived condition of all school assets, from tables and seats, to the structural state of classrooms and computer lab. This characteristic may be considered of medium importance. Its score corresponds to the quality of the school infrastructure: 1 (very low quality), 2 (low quality), 3 (medium quality), 4 (good quality) and 5 (very good quality).

- **RI:** the overall evaluation of the previous criteria – which corresponds to the ranking index of a school, obtained according to quation 1. In Table 2 a list of criteria weighted by the degree of importance will serve as a basis to define how much the school is disadvantaged.

Equation 1: Summary of criterion assessment (RI)

$$RI = \frac{\sum_{i=1}^{k}(N_i I_i)}{k} \; where \begin{cases} k: Total\ number\ of\ criteria \\ N: Criterion\ score \\ I: Importance\ or\ Weight \end{cases} \quad (1)$$

Table 2: Weight of each criterion used.

Description of criterion	I: Importance of criterion or Weight
Proportion of Computers for Each Student	3
Computer classes	2
Quality of Technology Resources	3
Quality of Infrastructure School	2

3.2 Research Questions

Two research questions (RQs) are of interest here:

- **RQ1**: Are there any performance differences between the learning levels of programming among students at schools of varying economic capabilities?

- **RQ2**: Does the teaching of programming contribute to students' motivation towards other subjects (and not only mathematics or logic)?

The following hypotheses are associated to **RQ1**:

Null hypothesis, 1H0: there is no difference between levels of programming (measured by the number of completed programming levels) among students from schools of different economic capabilities. The null hypothesis may be formalized as follows:

$$\mathbf{1}H_0: \mu_{favored\ schools} = \mu_{disadvantaged\ schools}$$

Where the notation μ_{school} represents the mean of the performance results (scores) by participating sudents from *school* in exams or tests after concluding the Introduction to Programming Course.

Using similar notation, one may define the other hypotheses as follows.

Alternative hypothesis, 1H1: there is a significant difference between levels of programming learning (measured by the number of completed levels) among students from schools of different economic capabilities. The alternative hypothesis is formalized as follows:

$$\mathbf{1}H_1: \mu_{favored\ schools} \neq \mu_{disadvantaged\ schools}$$

And for **RQ2**:

Null hypothesis, 2H0: teaching of programming does not contribute to student motivation towards other disciplines in general (not just math and logic). The null hypothesis is formalized as follows:

$$\mathbf{2}H_0: \mu_{motivation\ towards\ other\ disciplines} = \mu_{motivation\ in\ other\ post-course\ subjects}$$

Alternative hypothesis, 2H1: teaching of programming contributes to student motivation towards learning other disciplines (not just math and logic). This alternative hypothesis is formalized as follows:

$$\mathbf{2}H_1: \mu_{motivation\ in\ other\ disciplines} \neq \mu_{motivation\ in\ other\ post-course\ subjects}$$

For the development of the work, the underlying research is of mixed nature, i.e., the quantitative results from the case study with code.org in terms of score on tests on programming skills and knowledge that were acquired with the taught course are complemented with interviews with teachers from participating schools. These interviews served to

estimate students' post-course academic motivation and were conducted through telephone calls, based on open-ended questions (please refer to subsection 3.5).

3.3 Code.org Course Program

Code.org is a non-profit organization that provides a programming teaching platform with the goal of promoting and teaching programming to people of all ages. The site includes free coding classes and the initiative also targets schools in an attempt to encourage them to include more computer classes in their curricula. The main objective of a code.org course is to find solutions to problems that are presented graphically and the student are supposed to provide solutions in the form of a set of sequenced blocks, for example.

The founders of code.org believe that Computer Science should be a discipline alongside conventional disciplines such as mathematics, physics, and biology (Pantaleão, Amaral and Silva, 2017). For each new activity on Code.org there is always a video at its beginning, with the purpose of motivating and explaining how to carry out the activity. For children who do not know how to read, activities are presented as images and symbols which are linked to the daily life of the child.

An illustration of a course activity is given in Figure 1. In this activity, the student must assemble a sequence of blocks corresponding to the steps necessary for Angry Bird to reach the green pig.

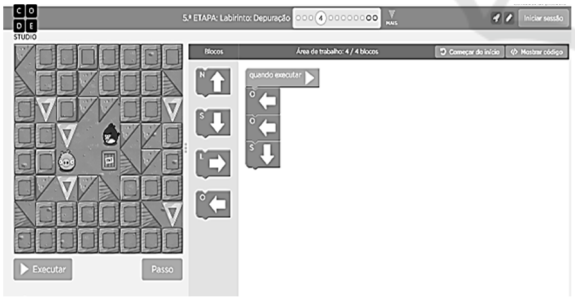

Figure 1: Code.org platform screen that shows the organization of a task to be solved by the student.

The methodology for applying the introduction to programming course consisted of five steps (as shown in Figure 2). The first one evolved around the planning of the basic concepts to be worked out by the students. The concepts were chosen according to the level of difficulty for learning, considering the age of the students and the respective school year they were at. The importance of learning these basic programming concepts for advanced programming was also considered. Five concepts were chosen in step one:

- **Algorithms:** An algorithm is a sequence of "actions" that must be followed by the computer to solve a certain task. This sequence of actions, specified by the human programmer, much like a cake recipe, will produce a solution for the problem at hand.

- **Sequencing:** a set of commands of an algorithm, which are organized in a sequence, in an attempt to solve a certain problem or task. On the Code.org platform, the user orders a sequence of instructions / actions by dragging blocks of commands and arranging them one below the other, thus forming the algorithm.

- **Repetition Block:** a set of blocks repeated in an algorithm. When the student reaches this learning stage, the alternatives will be studied to improve the produced algorithm, introducing blocks that replace the repeated blocks, reducing the size of the algorithm, but having the same purpose and results when executed.

- **Debugging Algorithms:** the process of finding and fixing defects of a particular algorithm or software. Typically, these defects produce an unexpected (wrong) result or do not execute when compiled. In the proposed course, some problems are presented to the student already with pre-defined solutions, but with errors. The student must debug the algorithm to find and fix the defects.

- **Condition Block:** a flow deviation control structure present in programming languages that performs different computations or actions depending on whether the condition is true or false. In the planned course, the "if-then (-soon)" structure was worked through blocks that represent this structure.

In the second stage, a mapping of the schools that could participate in the project in the city of Campina Grande-Paraíba, Brazil was carried out. Next, the mapped schools were visited and evaluated to check whether they had the necessary infrastructure (e.g. computers and internet access) to support the introduction to programming course. Given a school's infrastructure was considered adequate, the code.org based course was then offered for students at that school.

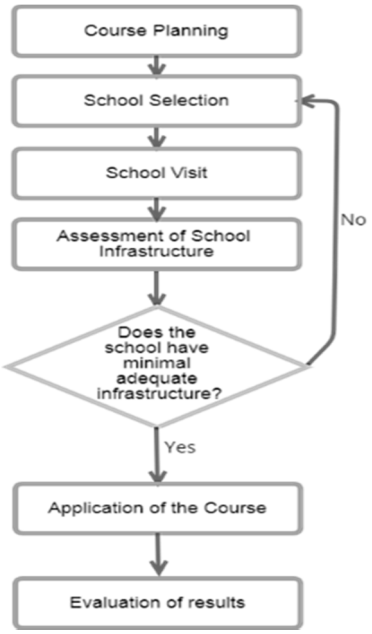

Figure 2: Methodological process for the courses.

3.4 Evaluation Procedure

After completing the course, students underwent a quantitative assessment using the data stored on the Code.org platform. In addition, a qualitative post-course questionnaire about their (new) academic behavior was answered by teachers of the participating schools. Indicators such as motivation and concentration were used as proxies for "academic behavior".

3.4.1 Quantitative Evaluation

When solving problems in the course based on the Code.org platform, two main metrics were collected: Number of Completed Levels and Lines of Code. The Number of Completed Levels corresponds to the number of problems solved during the course. Lines of Code corresponds to the amount of code written when solving problems in the course. Consequently and typically, the higher the number of completed levels, the greater the number of lines of code written. However, it should be noted that the number of lines of code may vary due to the solution used to solve a problem.

3.4.2 Post-course Qualitative Evaluation

After the course was taught at a school, the open questions in the questionnaire of Table 3 were used with the objective of verifying possible improvement in students' academic behavior. Such behavior was summarized by metrics that include: class attendance, study motivation, and participation in classes.

Table 3: Questionnaire for post-course evaluation.

Id	Questions
Q1	Do participating students' appear more motivated towards academic activities in general?
Q2	Has the aggressiveness (bullying) by students who took the course reduced?
Q3	Have students become more motivated to attend computer classes?
Q4	Has the concentration of students increased during class?
Q5	In general, did the class improve compared to classes that did not participate in the course?
Q6	Do students participate more in the classes after the course?

4 RESULTS AND ANALYSES

Courses on Intrduction to Programming based on the code.org plataform were carried out in six public schools (A to F) in the city of Campina Grande, and at one private school in the city of João Pessoa. Both citieis are in the state of Paraíba in Northeastern Brazil. On average, each class had 12 students who participated in the course, with a total workload of 3 hours. Participating students were typically 11 years old and were enrolled in the 5th year of elementary education. Each course lasted 3 hours and each lesson lasted for an hour and a half. The courses were taught by one of the authors of this paper. In some schools, existing IT teachers participated as assistants to the course's instructor and, at the same time, learned how to use the platform to continue this project in their schools. The research questions are addressed next.

Research Question 1: are there any performance differences between the learning levels of programming among students at schools of varying economic capabilities? Figure 3 and 4 offer some insght which might allow one to answer this RQ1 preliminarly.

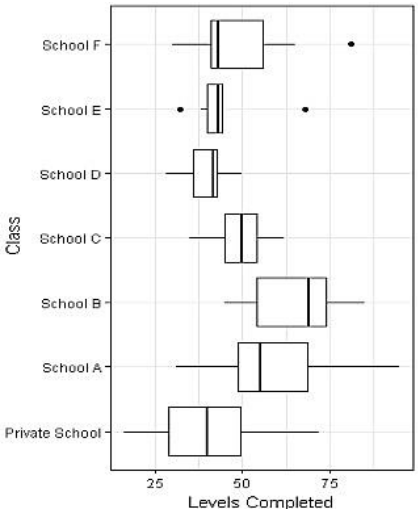

Figure 3: Boxplot of completed levels over all classes.

From Figure 3, one may observe that:

a) Classes from all participating schools, even from those with economic restrictions, finished the course with the Complete or Superior Levels completed. Despite having limited infrastructure and lower quality internet access, public school students achieved good results.

b) Two outliers with high values of completed programming levels (to the right in the figure) of classes from schools E and F signal students who were able to stand out and, during the short period of time the course lasted for, obtained higher levels of programming proficiency. On the other hand, the outlier with below-average value from school E is due to students who were later identified as having trouble reading or concentrating. Reading difficulties, being a direct (but not sole) responsability of educators, should be addressed as early as possible or the affected individuals may risk having their negative effects spread to other disciplines.

c) The most economically favored school of the sample – the Private School – does not show better results than its public, economically disadvantaged counterparts. Human resources - i.e., students in this case – may compensate for deficiencies in other resources, up to a point (one cannot do real programming without machines). Figure 4 may shed more light into this observation.

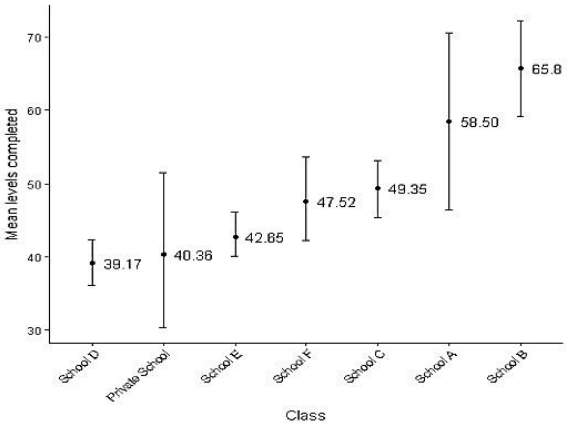

Figure 4: Averages of programming levels completed by classes (bars denote 95% confidence interval).

Figure 4 highlights the means of levels completed in each school, and their respective 95% confidence intervals. As can be observed, schools A, C, F, E and Private School present overlapping confidence intervals. Thus, it is not possible to isolate significant statistical differences among them. However, one can state with 95% certainty that School B has a significantly larger number of completed levels than the other schools, except for school A. Similarly, school E shows a significant difference from schools C and A, having a lower average compared to them.

It is important to note that the students of school B - even though they have fewer computers available and their school provides no regular computer classes - come out ahead in terms of better metrics than those of their peers at the other schools (except for school A whose 95% confidence interval's higher half overlaps with that for school B). This leads one to especulate that students, having come into contact with the novelty of learning programming, may have worked harder to explore the "unusual" opportunity. The quality of infrastructure and of the technological resources may have also contributed for the number of levels completed by B students. Note from Table 1 that school B, even though it has fewer computers per capita, presents better structural and equipment quality. This suggests that it pays to invest in quality – even in the case of a modest investment. Further, the 0.5 computer per student ration naturally led to "pair programming" at school B (i.e., two students per machine doing programming assignments). This agile programming (Abrahamsson et. al., 2017) recommended practice does indeed promote academic benefits in terms of discussing and adopting better solution alternatives.

One can also observe that according to Figure 4, there is no significant relationship between the more favored (or less disadvantaged) schools and the number of completed programming levels. With 95% confidence, the school with better resources (i.e., the private school) presents an average similar to that of its more disadvantaged counterparts.

Figure 4 suggests that students from too much economically disadvantaged or favored schools will do worse (operating in extremes is usually not a good strategy). Indeed, students from schools E and F and the private school show programming proficiency results that could be 20 to 40% less than their peers from other schools. It pays to invest (even a little) in quality for that may motivate students to excel, knowing they will be able to accomplish what they have set out to do. Motivation is further explored in RQ2.

Research Question 2: Does the teaching of programming contribute to students' motivation towards other subjects (and not only mathematics or logic)?

To answer RQ2, the post-course questionnaire of Table 3 was presented to eight teachers of the seven participating schools and yielded the following results. It is worth noting that a class at the studied schools usually has just one teacher, or at most two, who teaches all the curricular components of the school series. All interviewed teachers (six females; two males) had at the time, more than 5 years of experience teaching children and young people.

Interviewed teachers stated that students who took the introduction to programming course are more likely to participate in classes. Teachers also perceived a reduction in bullying (question 2 in Table 3) as the consolidation of answers in Figure 5 illustrates. Bullying is typically, a school problem (Oliveira et al., 2015). One may speculate that its causes include low (beneficial) occupation of pootential bullies. Interviewed teachers agree that students who took the programming course seem to become more focused and intrested on their studies. That may have led to the perception of reduced bullying at school. But that is not explicitly endorsed by (Oliveira et al., 2015) - although it may be covered in the indicated "other causes" - nor can one claim here to have produced evidence for such cause-effect relationship. For that, furhter and longer lasting studies are needed.

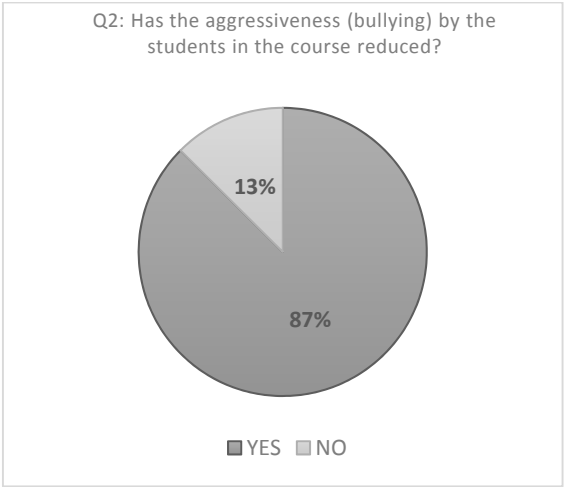

Figure 5: Qualitative research Q2 results.

Teachers also noticed that students have become more motivated to learn about computer science, regardless of the subject taught in the lab. Interest in programming has increased and students ask for classes on the Code.org Platform. Interviewed teachers said they noticed a greater concentration on academic activites by participating students particularly in regards to IT-related subjects (at schools where they are taught).

Regarding overall academic performance of participating students, the teachers commented improvements were across all subjects but more noticeable in mathematics. Teachers believe having introduction of programming as regular curricular discipline with a weekly workload in the computer lab will bring benefits to all involved.

The answers to the questionnaire and results in Figures 3 and 4 provide (early) evidence the designed and taught course will positively impact participating students' academic performance. For that there should be some investment in the quality of the school's supporting sinfrastructure and IT resources. Such an investment seems to cause better results overall, across the curriculum, even at economically disadvantaged schools. However, it is important to notice that, at the stage of the research reported in this paper, the post-evaluation discussed here was solely carried out with teachers. A more thorough and comprehensive longitudinal (i.e., across several subjects in the school curriculum) evaluation of participating students' gains with the programming course needs to be done in the future for sounder and further claims.

5 CONCLUSIONS AND FUTURE WORK

This article discussed the entire process for the creation and execution of an introduction course on programming applied in private and public schools, with the objective of answering two research questions: i) Are there any performance differences between the learning levels of programming among students at schools of varying economic capabilities? And, ii) Does the teaching of programming contribute to students' motivation towards other subjects (and not only mathematics or logic)?

Through the use of the code.org platform, metrics were collected that allowed exploration of correlations between student performance and issues related to school economic constraints. A qualitative post-course research was also carried out, with the objective of analyzing gains of motivation and performance of students in the classroom.

The results, both with the analysis of the data of the platform and with the post-course research carried out with the teachers, allowed one to consider some interesting aspects. In relation to the data obtained from the platform and with the methodology applied to classes, the students of public schools, even in an environment with budget constraints and inferior infrastructure, compared rather well against their peers from the private school with better economic conditions. This suggests that the platform environment allows for effective study, making students who never had contact with programming, solve problems presented during the course. In addition, the curiosity of the students of the public schools for novelty, that is, for being immersed in a course that they never had the opportunity before, favored their engagement and their good results at the end of the course.

The post-course research conducted with the teachers indicated that participating students became more motivated towards their studies and more participative and more concentrated in classes. These findings and possibly new ones will have to be verified further with new longer lasting validation studies, with the introduction of a course or even a programming discipline with a longer workload. The research also suggests that if larger investments are made in public schools, with the introduction of a programming discipline or for improvement of available laboratory infrastructure, their students may achieve better performance.

The project brought contributions to the school environment. Considering the results of public school classes, one may state that programming courses can be an important tool for digital inclusion in schools that have financial restrictions. In the scope of the research, contribution came about by showing that, applying a pedagogic methodology capable of inserting the student in an environment that arouses her or his curiosity and motivation, good results will result.

As future work, one can anticipate additional experiments to collect more meaningful statistics; to select topics, concepts and programming facilities to be included in course material; and to investigate their impact on the motivation of disadvantaged students towards education. Additional experiments may start with pilot projects to introduce the study of programming as a discipline in the curriculum.

REFERENCES

Abrahamsson, Pekka; Salo, Outi; Ronkainen, Jussi; Warsta, Juhani; Agile Software Development Methods: Review and Analysis. 2017. *Technical Research Centre of Finland.*

Barbosa, Fernanda de Arruda Campos. 2017. Computer Lessons in Early Childhood Education and Elementary Education: Importance and Benefits for Integral Formation. *Technological Interface Magazine*, pages 9-20.

Duarte, Clarice Seixas. 2007. Education as a Fundamental Law of Social Nature. Educ. Soc., pages 691-713.

Grivokostopoulou, Foteini; Perikos, Isodoros; Hatzilygeroudis, Ioannis. 2016. An Educational Game for Teaching Search Algorithms. In *Proceedings of the 8th International Conference on Computer Supported Education*, pages 129-136.

Grover, Shuchi, Roy Pea, and Stephen Cooper. 2016. Factors influencing computer science learning in middle school. *Proceedings of the 47th ACM technical symposium on computing science education*. ACM.

Ibáñez, María-Blanca; Di-Serio, Ángela; Delgado-Kloss, Carlos. 2014. Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. IEEE Transactions on Learning Technologies, pages 291-301.

Kalelioglu, Filiz. 2015. A new way of teaching programming skills to K-12 students: Code.org. Baskent University. *Computers in Human Behavior*, pages 200-210.

Oliveira, Wanderlei Abadio de et al. 2015. The causes of bullying: results from the National Survey of School Health (PeNSE). *Revista latino-americana de enfermagem,* pages 275-282.

Pagani, Mario Mecenas. 2015. The Insertion of Technologies in Luddical and Recreational Activities

in the 4TH and 5TH Years of Fundamental Teaching. *FAEMA Scientific Journal,* pages 90-106.

Pantaleão, Eliana; Amaral, Laurence Rodrigues; Silva, Glaucia Braga. 2015. An approach based on Robocode environment for teaching programming in high school. *Brazilian Journal of Computers in Education,* Vol 25.

Papadakis, S.; Kalogiannakis, M.; Orfanakis V.; and Zaranis, N. 2017. The appropriateness of scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, 12(4), 58-77.

Pelgrum, W. J. 2001. Obstacles to the integration of ICT in education: results from a worldwide educational assessment. *Computers & education*, 37(2), 163-178.

Pina, Alfredo; Rubio, Gabriel. 2017. Using Educational Robotics with Primary Level Students (6-12 Years Old) in Different Scholar Scenarios: Learned Lessons. In *Proceedings of the 9th International Conference on Computer Supported Education*, pages 196-208.

Rapkiewicz, Clevi Elena; Falkembach, Gilse; Seixas, Louise; Rosa, Núbia dos Santos; Cunha, Vanildes Vieira; Klemann, Miriam. 2006. Pedagogical Strategies in The Teaching of Algorithms and Programming Associated with The Use of Educational Games. *New Technologies in Education Magazine*, pages 1-11.

Rodrigues, M. C. 2002. How to Teach Programming? *Informatics – Newsletter,* Year I n° 01, ULBRA.

Silva, Teresa Lúcia. 2009. The financing resources decentralization as a democractic management inductor. Study about the public schools in São Carlos. *Digital Library of Theses and Dissertations of USP*.

Sung, HY; & Hwang, GJ. 2013. A collaborative game - based learning approach to improving students learning performance in science courses. *Computers & Education*, pages 43-51.