ISSN: 2766-9823 Volume 3, 2021

The impact of practicing the computational thinking in a programming course

Anabela Gomes
Coimbra Institute of Engineering
Polythecnic Institute of Coimbra
CISUC - Center for Informatics and Systems of the University of Coimbra
Coimbra, Portugal
anabela@isec.pt

Fernanda Brito Correia
Coimbra Institute of Engineering
Polythecnic Institute of Coimbra
Coimbra, Portugal
fernanda@isec.pt

E. Bigotte de Almeida
Coimbra Institute of Engineering
Polythecnic Institute of Coimbra
CASPAE-Centro de Apoio de Pais e Amigos da Escola
Coimbra, Portugal
ebigotte@isec.pt

Ricardo Almeida
CASPAE-Centro de Apoio de Pais e Amigos da Escola
Coimbra, Portugal
ricardo.almeida@caspae.pt

Abstract— **Engineering students have problems related to problem-solving skills, which reflects on a high failure rate in introductory programming curricular units they attend. To try to reduce this failure rate and to help students to acquire the necessary programming competences, new processes of teaching and learning are needed. Plans should be drawn to foster positive attitudes towards programming to reinforce students' motivation to learn and to apply their knowledge to new situations. In higher education and from an early stage, constructing instruments that facilitate the learning and teaching process and the promotion of student involvement, contributes to build a sustainable structure that includes students' projects carried out in areas of programming knowledge, which are considered essential to facilitate the assimilation of computational thinking. To achieve this, it was implemented a collaborative project, where students should develop a project using the Scratch software. This collaborative project was evaluated, using several questionnaires with various statements answered by the students after they finished their project. These questionnaires were analysed, by grouping the statements of the questionnaires into categories and sub-categories. Conclusions, about the usefulness of mapping both Scratch and C programming language in the process of learning and teaching, in the learning of programming introductory concepts and in the students' motivation to try other programming languages, are extracted from the evaluation of their projects and from the analysis of these questionnaires.**

Keywords— Scratch, Higher Education, Mathematics, Programming, Computational Thinking Skills

I. INTRODUCTION

Higher Education institutions with Engineering degrees have a very heterogeneous public with diverse personal, motivational and cognitive characteristics and several knowledge backgrounds. The literature includes many references about the high failure and dropout rates in introductory programming courses in many higher education institutions worldwide [1-6]. Many causes for the learning difficulties associated to these courses have already been identified. Usually students have lack of preparation in mathematics and problem solving [7-10].

In order to contemplate this problem, every year, teachers try to reflect about the use of different and innovative teaching methods. They look for new methodologies and strategies that can motivate students in order to obtain better learnings and better results, while maintaining the level of demanding and objectives.

New students arrive to engineering degrees with different levels of problem-solving knowledge. This happens majorly because their previous courses are less focused on the autonomous resolution of problems and logical reasoning. Such problem has transversal repercussions to several courses, like those related with mathematics and programming [11-12].

As Biomedical Engineering teachers we are naturally sensitive to this problem. It is well known that these students have problems related with problem-solving skills, which reflects on the results to Programming Introduction course. Consequently, in the 1st semester from the 1st year, a higher failure rate is noticed in the 1st programming course they attend. It is also well known that these students are highly motivated for areas and topics that use study methodologies through rote learning and mechanization, such as health topics in detriment of technologies.

The assumptions of the Bologna model and the recommendations of the Agency for Evaluation and Accreditation of Higher Education (A3Es) point out precisely in the sense that there must be a higher concern in the planning of transversal competency training throughout the course. Thus, it is considered that the development of multidisciplinary projects may be a relevant strategy for training such students and also for the integration of students in higher education.

Therefore, teachers from two curricular units from the 1st year and 1st semester, one about introductory programming, Introduction to Information Technology (ITI), and another about mathematics, Integral Calculus (IC), decided to carry out a collaborative project that involved their common students. This project allows, on the one hand, the student's development of programming skills, and on the other hand, the involvement of all participants in the development of mathematical pedagogical resources. This collaborative project had two main goals to achieve. As a 1st goal, to allow the development of computational thinking skills through the use of different learning strategies to help the learning process of the students involved in the various curricular units of their Biomedical Engineering degree, in particular in the curricular units that contain programming subjects. As a 2nd goal, to involve students in the construction of mathematical pedagogical resources. The construction of these resources will also allow the development of abstract thinking and the ability to modularize, reducing a problem to simpler situations, representing problems in different ways, making analogies with similar problems and developing deductive thinking. This process allows the improvement of several problem-solving skills through the expansion of analytical, quantitative, analogical and combinatorial reasoning skills.

In this context and with this approach of an interdisciplinary project, the authors decided to use the Scratch software as a tool, for students to learn how to develop projects effectively with different levels of complexity, learning the logic of programming and how to solve problems, and at the same time develop their autonomy of computational thinking.

Scratch software allows to learn mathematical basic concepts in a constructive manner and awakens student interest and motivation to program [13], developing in students their own path for learning. Felleisen and Krishnamurthi [14] emphasized the concept of "imaginative programming", considering its importance as an element of programming referring the close relationship between mathematics and computation.

II. The Study

The study focused on students of the Biomedical Engineering degree, ministered at the Coimbra Institute of Engineering (ISEC) from the Polytechnic Institute of Coimbra (IPC), Portugal, enrolled in the courses of ITI and IC, in the 1st semester, 1st year. The sample is composed by 33 students (11 males and 22 females), where 72.4% are between the ages of 18 and 20 years. The subjects contained in ITI are the ones usually taught in a first programming course. Here we use the C language to teach the fundamental procedural programming concepts. The subjects covered include the basic concepts taught in an introductory programming course using a procedural programming language, like data types, operators and expressions, standard input and output formatted data, data structures, functions, arrays and string manipulation. In the first three weeks students solve programming problems using sequential, selection and repetitive structures through pseudocode. Only after a certain comprehension of the subject the problems are solved using the C language giving emphasis to the syntax details. This course has 4 contact hours per week, 1 hour of theoretic class and 3 hours of lab classes in two groups of about 30 students each. In addition, teachers offer more 6 hours per week to clarify students' doubts, during the entire semester.

Due to the already mentioned difficulties of the students to learn programming we have decided to use the Scratch tool to help them to learn the programming concepts and structures, since this tool allows a higher abstraction level and a visual approach of programming. To help students to overcome their difficulties and have better programming results, we conducted an interdisciplinary project developed as a curriculum complement of ITI with the collaboration of IC, since students have both curricular units. After the conclusion of the project and considering the main objectives of our investigation we tried, using qualitative and quantitative approaches, to understand the perceptions of students in relation to the importance of this project.

For this analysis we studied two aspects. Firstly, in relation to the use of Scratch software. Secondly, to validate the importance of undertaking the developed project to develop computational thinking and programming learning.

This collaborative project consisted firstly about raising awareness on the Scratch platform: how it should be used, what its potentialities are, what students can and cannot develop, and how they can do it. For the multidisciplinary team, it is a phase of diagnostic evaluation, perceiving the students' major difficulties, the areas that should be more worked, and the projects they like to develop, where their motivation is greater. After this, there was a learning phase of the platform, where students learned to develop projects effectively with different levels of complexity, including the leaning and practice of the logic of programming, and how to solve problems while developing their autonomy of learning. Students had full autonomy to build a project, following previously established guidelines, in groups of 2 elements, to be presented at the end of the semester. Teachers and a monitor offered introductory workshops and accompanying extra classes' hours, intervening only when requested by the students, providing autonomous learning based on trial and error. The various projects developed by the students were educational games or animations inspired by introductory mathematical themes. Finally, students had to prepare a final report and a final presentation.

To evaluate this collaborative project, teachers developed several questionnaires. Here two questionnaires are analysed: A1 – a questionnaire about the satisfaction, importance and difficulties of the students when using Scratch. This questionnaire also includes questions related to computational thinking skills. A2 – a questionnaire about the impact of Scratch used in programming learning. In each questionnaire, the format of the scale used was a five-point Likert type, which describes the student degree of agreement, where 1 was the minimum value and 5 was the maximum value. Being 1 – Totally disagree, 2- Disagree, 3 – Do not agree or disagree, 4 – Agree and 5 – Totally agree.

The questionnaire A1 has 18 statements. The 1st 12 statements we categorized in the following three categories: "Satisfaction", "Utility and ease of access" and "Ease of learning" (Table1). The remaining 6 statements in the A1 questionnaire were related to computational thinking, inspired in part by the computational thinking grid [15]. There were statements related with the following aspects "Experiment and Interact", "Test and Correct" and "Reuse and recombine" (Table 2). The category "Abstracting and Modularizing", was not contemplated with a direct question but taken out through a more detailed topic that students had to include in the project report. The other categories were also supplemented with descriptions required in the mentioned report. For example, on "Experimenting and Interacting" they had to indicate how they built the project in the most detailed way as possible. About "Test and Correct", in the report, students also had to indicate a section where they should describe the main problems they had and the way they proceeded to solve them. About "Abstracting and Modularizing", students were asked to indicate in the report how it was decided which sprites were needed for the project and where they were used and for what purpose.

TABLE I. USE OF SCRATCH TOOL

Category	Question
Satisfaction	A1Q1. I liked to work with the Scratch tool.
	A1Q3. I felt motivated to work with Scratch.
	A1Q17. I would like to work with Scratch in the future.
Utility and accessibility	A1Q4. Scratch is an ideal tool to build educational materials.
	A1Q5. The Scratch tool is of easy access.
	A1Q15. I recognize in Scratch different potential.
	A1Q18. I recognize in Scratch important features for those who start in programming.
	A1Q19. I understand that it is important for all graduates in Biomedical Engineering receive training in Scratch
Ease of learning	A1Q2. I felt difficulties when using the Scratch tool.
	A1Q6. The learning process of the Scratch tool was easy.
	A1Q7. The programming environment is easy to understand and use.
	A1Q8. Generally, I felt comfortable to use the Scratch tool.

TABLE II. COMPUTATIONAL THINKING

Category	Question
Experiment and interact	A1Q9. I reviewed several times my project.
Test and correct	A1Q10. The final result of my project was the intended.
Test and correct	A1Q11. I easily identified code errors, when something wasn't right.
Reuse and recombine	A1Q12. I tried to program other projects during the preparation of the main project.
	A1Q13. I researched other Scratch projects to inspire me.
	A1Q14. I understood the code from other projects and used these codes in my project.

The questionnaire A2 had 7 statements that we categorized in the following four categories: “Concepts”, “Scope/Range”, “Algorithms” and “Active learning/autonomy”. The category “Algorithms” was divided in 4 subcategories: “Specification”, “Analysis”, “Processing” and “Test” (Table3).

TABLE III. IMPACT OF SCRATCH USE IN ITI

Category	Sub-category	Question
Concepts (selections, repetitions, …)		A2Q1. Better understand certain concepts of programming.
Scope/range		A2Q2. Know the type of questions and answers that programming can and cannot provide.
Algorithmics	*Specification*	A2Q5. Identify and specify programming problems.
	Analysis	A2Q4. Recognize the central ideas necessary in solving a programming problem.
	Processing	A2Q3. Understand and evaluate a set of logical steps required in an algorithm.
	Test	A2Q6. Solving programming problems including the application of appropriate algorithms.
Active learning/autonomy		A2Q7.Develop active learning processes (the student contributes to his learning process actively) in the field of programming.

The questionnaires were answered by the students, who performed a public presentation and assessment of their final work, with a joint and shared debate of each project developed by them. In order to develop complementary studies in the context of the contribution of this action for the improvement of the performance of the students in order to allow a better understanding of each student trajectory, student identification was requested, but not mandatory. The confidentiality in the treatment and disclosure of the data was guaranteed.

III. THE RESULTS

This section reports the obtained results after the questionnaires (A1 and A2) analysis. Here are also mentioned aspects considered relevant and extracted from the students' final project report.

A. Scratch impact analysis

Figures Fig. 1, Fig. 2 and Fig. 3 show the graphics of the quantitative analysis of A1 questionnaire questions concerning the categories “Satisfaction”, “Ease of learning” and “Utility and ease of access” regarding Scratch tool used in students' projects.

Concerning the 1st sub-category “Satisfaction”, about the satisfaction using Scratch software, (A1Q1), in general, the involved students liked to work with it, with 67.65% of the sample agreeing with the statement and 26.47% fully agreeing. When asked about motivation (A1Q3), 73.53% of students agreed that they felt motivated to work with Scratch. They also refer that they would like to work with Scratch in the future, in a professional context (A1Q3) (I agree - 47.06% and I totally agree - 11.76%).These results seem to indicate that even though students refer that they were not so motivated, they liked working with Scratch.

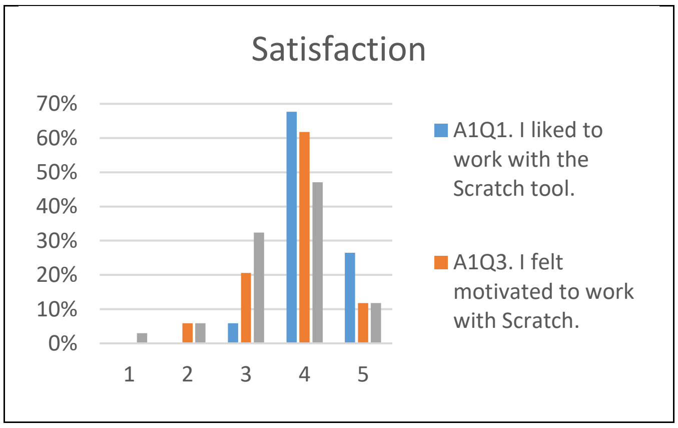

Fig. 1. Satisfaction of using Scratch.

However, the overwhelming majority (A1Q4: I Agree - 44.12% and I Agree Totally - 47.06%) found it to be very useful for building educational materials while simultaneously being easily accessible (A1Q5: Agree - 35.29% and Totally Agree - 55.88%). They also recognize in Scratch different potentialities (A1Q15: Agree - 50.00% and Totally Agree - 35.29%), and important characteristics for those who start programming (A1Q18: Agree - 44.12% and Totally Agree - 41,18%), understanding that it is important for all graduates in Biomedical Engineering to receive training in Scratch programming when they start programming (A1Q19: Agree - 50.00% and Totally agree - 8.82%).

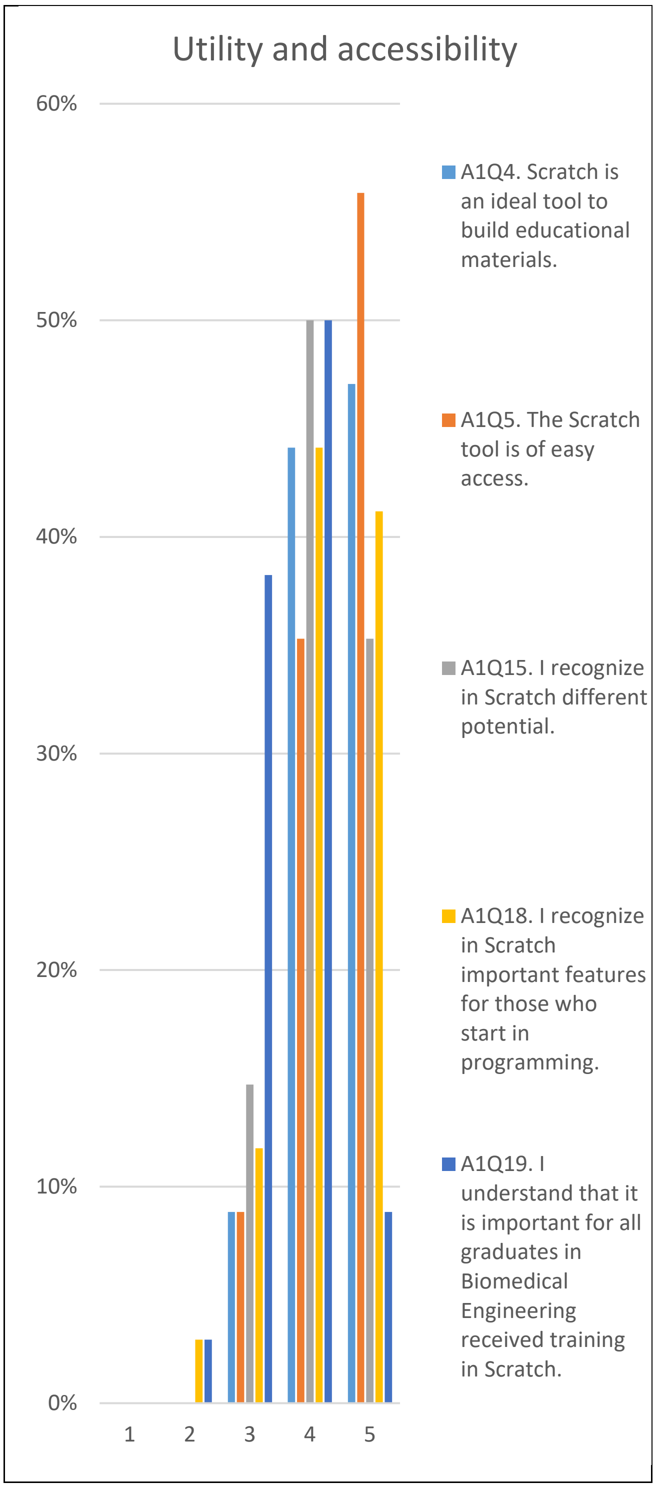

Fig. 2. Utility and accessibility of using Scratch.

Concerning Scratch software ease of learning, most students reveal that the learning process is easy (A1Q6: Agree - 61.76% and Totally agree - 11.76%), however, when asked about their difficulties using this tool in a specific situation the results were not so favourable (A1Q2 Agree – 41.18% and Totally agree - 2.94%). Students found Scratch's work environment easy to use (A1Q7: Agree - 58.82% and Completely Agree - 17.65%) and they generally feel comfortable using the tool (A1Q8: Agree - 64.71% and Totally agree - 14.71 %).

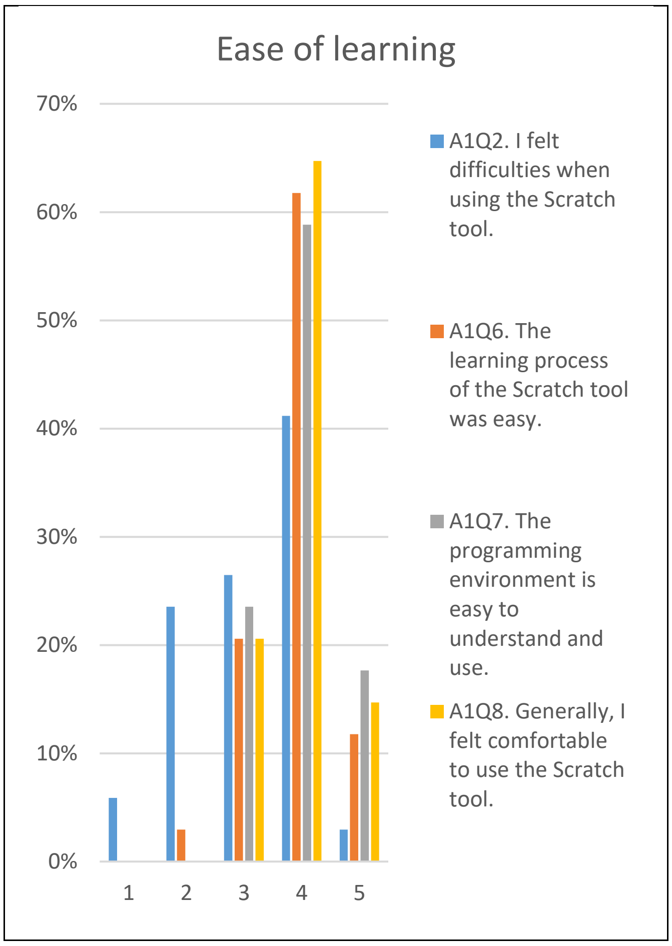

Fig. 3. Ease of learning of using Scratch.

B. Computational Thinking Analysis

As mentioned before, we also analysed the following dimensions of Computational Thinking, namely the sub-categories: “Experimenting and Interacting”, “Test and Correct”, “Reuse and Recombine” and “Abstract and Modularize”. Regarding the competence of the sub-category “Experimenting and Interacting”, students were asked to describe, in each project report, how they built their project, step by step in the most detailed way possible. However, in general, these descriptions were very elementary, not detailing specific aspects of it. Few gave a general description of the project, in an orderly manner. Still on this topic, students in A1Q9 were asked whether they had reviewed the project several times, with a significant majority (97.06%) of the students expressing agreement (partial agreement (38.24%) or total agreement (58.82%)) regarding having reviewed their project several times.

Regarding the competence of the sub-category “Test and Correct” (A1Q10 + A1Q11) we asked students if they easily identified errors in the code (A1Q11) and if the final result of the work was what was expected (A1Q10). Regarding the statement A1Q11, the majority (58.82%) of the students expressed agreement (partial agreement (52.94%) or total agreement (5.88%)) due to having easily identified error in the codes, only 5.88% did not partially agree and 35.29% having no opinion (neither agreed nor disagreed). Concerning the question A1Q10, the majority (76.47%) of the students expressed agreement (partial agreement (55.88%) or total agreement (20.59%)) regarding satisfaction with the final result of their project, 5.88% did not partially agree and having some students (17.65%) without an opinion (neither agreed nor disagreed). In the report, students also had to fill in a section where they should describe the main problems they had and how they proceeded to correct them. However, most students either did not describe the problems or were very vague in their answers about the problems, so we considered their answers to be of no relevance to this analysis.

Regarding the competence of the sub-category "Reuse and Recombine" (A1Q12, A1Q13 and A1Q14) we asked if they researched other projects (A1Q13), if they understood the code of other projects and if they used them (A1Q14) and if they tried to implement other projects (A1Q12). Regarding the question, A1Q13, the majority (82.35%) of the students expressed agreement, meaning that the majority researched other projects (partial agreement (58.82%) or total agreement (23.53%)), only 11.76% did not fully or partially agree, with 5.88% without opinion (neither agreed nor disagreed). Regarding the question A1Q14, the majority (70.59%) of the students expressed agreement, having understood the code of other projects and using it in their project (partial agreement (50.00%) or total agreement (20.59%)). 17.65% did not partially or totally agreed, with fewer students (11.76%) without an opinion. Regarding the question A1Q12, there was a percentage of 35.29% students who expressed agreement regarding having tried to program other projects during the elaboration of their project (partial agreement (23.53%) or total agreement (11.76%)). 38.24% did not agree partially or totally, meaning that they just did their project, with 27.47% of students having no opinion (neither agreed nor disagreed). Concerning this aspect, it was noticed that the majority of students tried to find inspiration in other projects, especially in coding approaches that could be useful for their project. Many of them deviated from the initial idea because they were unable to implement it or because they did not find available implementations in relation to what they had in mind. This information was deduced during the presentations and defences of the projects and also during the various phases of presentation of the project that deviated from what was initially intended. Those who tried to incorporate parts of other available projects, found it difficult to describe how they adapted ideas, scripts or resources from other projects. Worse than that, was the fact that they did not identify the sources and the authors in which they were inspired.

Regarding the competence of "Abstract and modularize", students in the report were asked to indicate how they decided which sprites were needed for the project and where they were used and for what purpose. However, the majority of the students provided little relevant or low-level descriptions, giving no general idea of the decision to choose certain sprites according to the general objectives of their project.

It should be noted that the works produced by students were also evaluated by the teachers and submitted to the Dr. Scratch tool [16]. Although we cannot draw conclusions with statistical significance given the sample size, we can make the following considerations. The results of the produced projects were generally good, being the average 82%, the minimum 66%, the maximum 100% and the standard deviation of 10%. Scratch helped students in terms of algorithms, helping to realize a set of programming concepts. Regarding the evaluation by Dr. Scratch, the differences were minimal with respect to the parameters under consideration, namely, "Flow Control", "Data Representation", "Abstraction", "User Interactivity", "Synchronization"," Parallelism "and "Logic". So, we did a correlation analysis between the global score (adding these parameters) obtained in each project and the respective students classifications, with no correlation being obtained. It was not a surprise, since we took into account in the project evaluation other aspects that stood out such as usability, pedagogical issues, presentation and defence, not related to the aspects evaluated by Dr. Scratch.

It was also clear, that this analysis involved only the teachers, so we consider that an evaluation and testing of projects developed by students, would be an added value to have in consideration in future projects.

C. Programming Knowledge analysis

Students' opinion about the impact of using Scratch in programming learning in ITI, A2 questionnaire, is illustrated in the graphics of Fig. 4, Fig. 5, Fig. 6 and Fig. 7.

Regarding the programming concepts understanding, A2Q1, the majority (57.69%) agreed, 46.15% partially agreed and 11.54% totally agreed that the developed project helped them understanding these concepts. Relatively to the scope and range of programming-related issues, A2Q2, 65.38% considered that Scratch's work gave them a better understanding of the type of programming-related questions and answers, although only 7.69 % fully agreed.

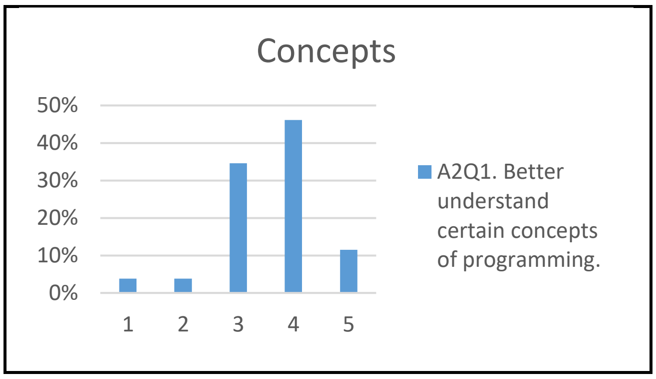

Fig. 4. Impact of the Scratch use in concepts programming learning.

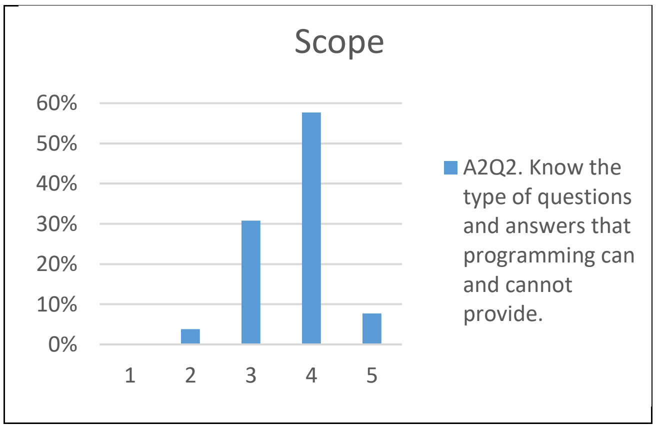

Fig. 5. Impact of the Scratch use in scope of programming learning.

With respect to the problem-solving ability of a programming problem as far as the algorithm is concerned, sub-category “Algorithmics” A2Q3 to A2Q6, most students answered that they agreed that their project helped them to develop this ability. To A2Q5 question, concerning the identification and specification of programming problems using Scratch, 73.08% of the sample agreed with the sentence (57.69% of partial form and 15.38% of total form). Regarding A2Q4 statement, asking if the work done in Scratch helped to better analyse and recognize the main ideas for solving a programming problem, the answers showed 57.69% of agreement (50% partially and 7.69% overall). The A2Q3 statement, if Scratch helped to better perform algorithmic processing with regard to the understanding and evaluation of the procedure and writing of the algorithm (i. e., the logical steps required to transform the input into output) gathered 84.62% of agreement (11.54% of partial form and 73.08% of total form). The A2Q6 statement, concerning if the work performed in Scratch helped to better test the adequacy of the algorithm used in solving a programming problem met 57.69% of agreement (11.54% of partial form and 46.15% of total form). It is notable that in these questions’ majority the percentage of students who did not agree was 0% and nearly 0% those who did not agree partially (3.85% in A2Q3, 7.69% in A2Q4, 0% in A2Q5 and 0% in A2Q6). This data can mean that the remaining percentages referred to students who stated that they had no opinion on the subject.

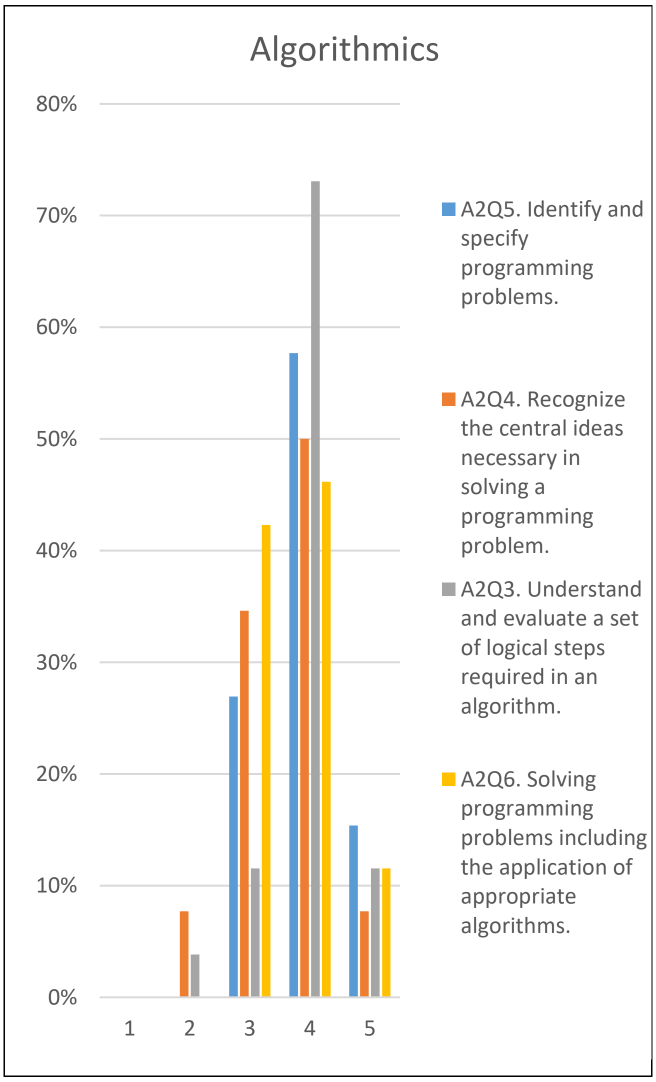

Fig. 6. Impact of the Scratch use in algorithmics programming learning.

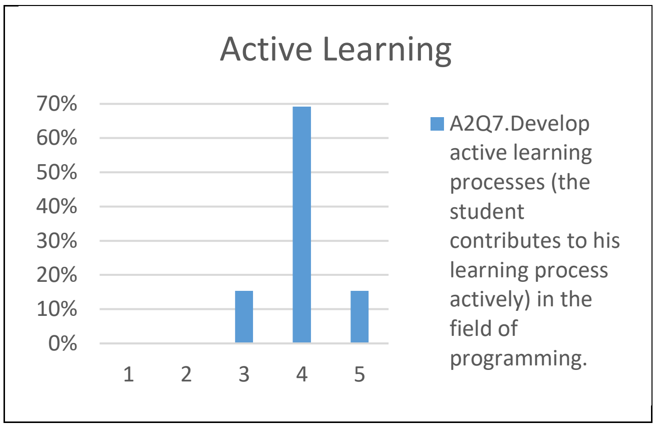

Fig. 7. Impact of the Scratch use in active programming learning.

When questioned about the advantages of carrying out their project, the obtained answers were, in general, very positive. They refer that their project helped them to improve the logical reasoning (useful in mathematics, programming and in many other engineering subjects), to improve the comprehension level (of programming concepts) and the skill to transmit and explain it to others due to the consolidation of acquired knowledge and the necessary organization level to plan larger tasks. They also mention that they acquired a set of competences necessary to learn programming effectively (logical reasoning, code organization, planning of major projects and consolidation of concepts, among others). They found Scratch useful to help them understand the concepts and realize its real meaning and application in concrete and real situations making it easy to adapt and transfer them to other languages more complex such as the C language. They felt more confident to study and try other programming languages due to the mapping using these two programming languages. They also referred that this project gave them a different perspective of the usefulness of programming with utility and fun. This triggered the desire to apply the learned concepts to new situations and make new projects, programming and developing games and new interactive applications.

About the disadvantages in carrying out these projects the obtained responses were essentially reported in relation to the time spent on each project. They reported that the work volume was excessive, requiring lots of time, dedication and persistence, forcing them to a hard work of thinking logically and found it much more difficult than all the other subjects they studied. The majority of students considered that, apart from these aspects there were no disadvantages. However, a minority considered that it was not so useful, mainly because this is not a programming language that they can use at a future professional work. Moreover, many were disappointed with the final result obtained regarding to all the effort applied. Some of them also pointed as a negative aspect the classes schedule for this project (done extra regular hours), namely that they worked too late and after a full day of classes it was difficult to find concentration to work.

A minority of the students also considered that Scratch project did not help them to understand C programming language. They also did not considered Scratch very useful to diversify the type of math exercises.

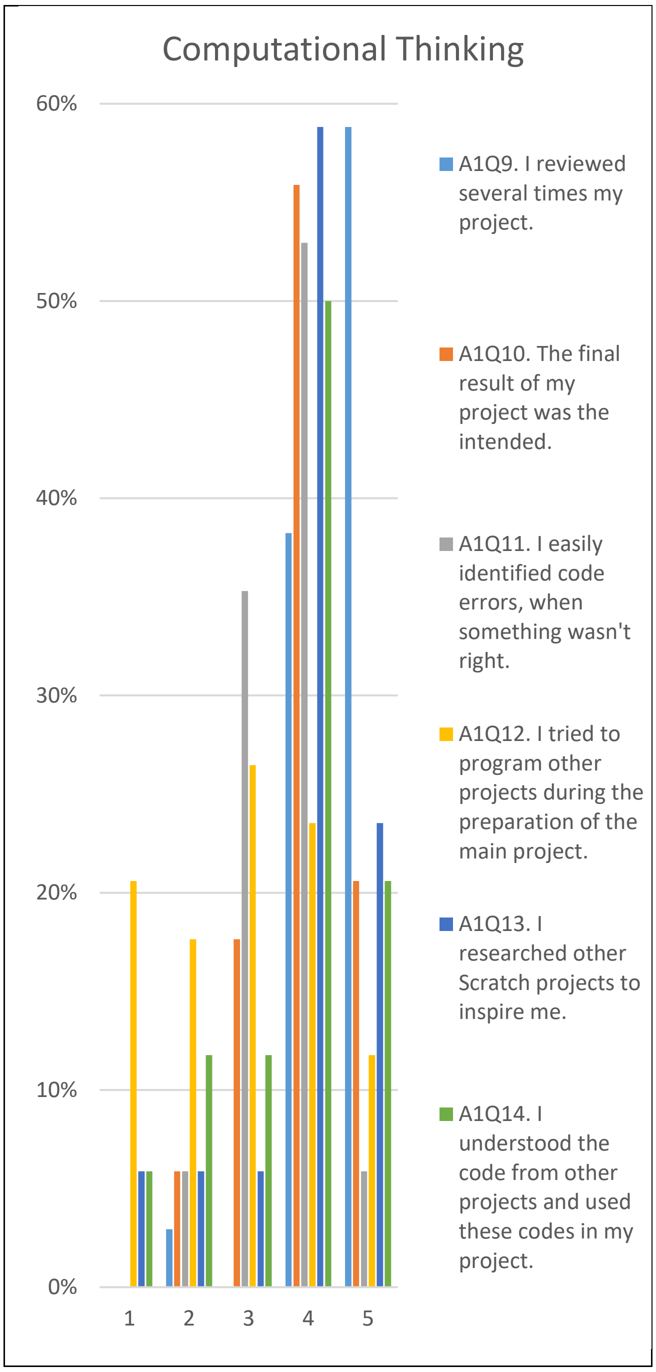

Fig. 8. Computational thinking with Scratch.

IV. CONCLUSIONS

As Biomedical Engineering teachers we are naturally sensitive to the difficulties of the students to learn programming. It is well known that these students have problems related with problem-solving skills, which reflects on the results of programming introduction curricular unit. In the 1st semester from the 1st year, a higher failure rate is noticed in the 1st programming curricular unit they attend. To try to reduce this failure rate and to help students to acquire the necessary programming competences, it was implemented a collaborative project using the Scratch software.

To evaluate the mentioned project addressed to students from the 1st semester from the 1st year, of the 1st programming curricular unit of Biomedical Engineering degree, teachers applied and analysed two questionnaires, answered by the students after developing their projects using the Scratch software, A1 and A2. A1 – a questionnaire about the satisfaction, importance and difficulties of the students when using Scratch. This questionnaire also includes questions related to computational thinking skills. A2 – a questionnaire about the impact of Scratch used in programming learning.

The questionnaires made of several statements and answered by the students using a 5-level scale were organized in categories and some of the categories in subcategories described in Table 1 and Table 2 for questionnaire A1 and Table3 for questionnaire A2. Besides that, students pointed out advantages and disadvantages of using the Scratch software to help them leaning how to program. The results seem to indicate that even though students refer that they were not so motivated, they liked working with Scratch. Most students reveal that the general learning process was easy, that they found Scratch's work environment easy to use and generally they felt comfortable using this software, but many students felt difficulties when using it in a specific situation.

Several dimensions of Computational Thinking were also analysed. The students achieved low values for the various parameters analysed in the computational thinking grid. This highlights the need for approaches that develop the computational thinking, so necessary for learning programming.

It was also done a correlation analysis between the global score, adding Dr. Scratch parameters obtained in each project, and the respective students classifications, but no correlation was found. A reason for that could be because it was taken into account, in each students' project evaluation other aspects, behind the aspects evaluated by Dr. Scratch.

Students also referred several advantages in developing a project using Scratch, like improving the logical reasoning, improving the programming concepts comprehension level or the acquisition of skills to transmit and explain the acquired knowledge to others. Most students, after finishing their project, felt more confident to study and try other programming languages due to the mapping using these two programming languages Scratch and C language.

About the impact of using Scratch in programming learning, the majority of students agreed that the developed project helped to understand programming concepts and gave those students a better understanding of the type of programming-related questions and answers. They also considered that their project helped to develop their algorithm problem-solving ability when programming.

As disadvantages students highlight that the work volume necessary to develop their project was excessive, which is understandable, since they had extra hours to develop and be accompanied in their project. Only a minority of students considered that their Scratch project did not help them to understand C programming language.

We also consider that the evaluation and testing of projects developed by students, besides professors, would be an added value to have in consideration in future projects.

ACKNOWLEDGMENT

The authors would like to thank all students that participated in the studies and to the teachers and people that allowed their realization.

References

[1] J. Bennedsen and M. E. Caspersen, “Failure rates in introductory programming,” *ACM SIGCSE Bull.*, vol. 39, no. 2, pp. 32, Jun. 2007, doi: 10.1145/1272848.1272879.

[2] C. Watson and F. W. B. Li, “Failure rates in introductory programming revisited,” in *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE ’14*, 2014, pp. 39–44, doi: 10.1145/2591708.2591749.

[3] T. Jenkins, “On the difficulty of learning to program,” in *Proceedings of the 3rd Annual LTSN_ICS Conference* (Loughborough University, United Kingdom, August 27-29, 2002), The Higher Education Academy, 2002, pp. 53-58.

[4] E. Lahtinen, K. Ala-Mutka and H-M Järvinen, H-M., "A study of difficulties of novice programmers," in *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Monte de Caparica, Portugal, June 27-29, 2005), ACM New York, NY, USA, 2005, pp. 4-18.

[5] R. Lister, B. Simon, E. Thompson, J. L. Whalley and C. Prasad, “Not seeing the forest for the trees: novice programmers and the SOLO taxonomy,” *SIGCSE Bulletin*, vol. 38, no. 3, pp. 118-122, 2006.

[6] P. Byrne and G. Lyons, “The effect of student attributes on success in programming,” *SIGCSE Bulletin*, vol. 33, no. 3, pp. 49-52, 2001.

[7] I. Milne and G. Rowe, “Difficulties in learning and teaching programming – views of students and tutors,” *Education and Information Technologies*, vol. 7, no. 1, pp. 55–66, 2002.

[8] J. Bennedsen and M. E. Caspersen, “Abstraction ability as an indicator of success for learning object-oriented programming?,” *SIGCSE Bulletin inroads*, vol. 38, no. 2, pp. 39-43, 2006.

[9] W. D. Gray, N. C. Goldberg and S. A. Byrnes, “Novices and programming: Merely a difficult subject (why?) or a means to mastering metacognitive skills?,” *Journal of Educational Research on Computers*, vol. 9, no. 1, pp. 131-140, 2007.

[10] J. Stachel, D. Marghitu, T. Brahim, R. Sims, L. Reynolds and V. Czelusniak, “Managing Cognitive Load in Introductory Programming Courses: A Cognitive Aware Scaffolding Tool,” *Journal of Integrated Design and Process Science, Computer Science*, vol. 17, no. 1, pp. 37-54, 2013.

[11] A. Gomes, L. Carmo, M. E. Almeida and A. J. Mendes, “Mathematics and programming problem solving,” in *Proceedings of 3rd E-Learning Conference – Computer Science Education* (Coimbra, Portugal, 2006).

[12] A. Gomes and A. J. Mendes, “A study on student performance in first year CS courses,” in *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education – ITiCSE’10* (Ankara, Turquia, 2010).

[13] L. Calao, J. Moreno-Leon, H. Correa, G.Robles, ”Developing Mathematical Thinking with Scratch An Experiment with 6th Grade Students”, G. Conole et al. (Eds.): EC-TEL 2015, LNCS 9307, pp. 17–27, 2015.

[14] M. Felleisen, S. Krishnamurthi, “Safety in Programming Languages,” in *Proceedings of the Sixth ACM-SIGSOFT Symposium on the Foundations of Software Engineering*, November 1998.

[15] K. Brennan and M. Resnick, “New frameworks for studying and assessing the development of computational thinking,” *American Educational Research Association meeting*, pp. 1-25. Vancouver: AERA.

[16] J. Moreno-León, G. Robles, G. and M. Román-González, “Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking,” *Revista de Educación a Distancia*, vol. 46, pp. 1-23.