Exploring a Hybrid Methodology: Experience Report in Introductory Programming for Computer Science and Information Systems Courses

Marluce Rodrigues Pereira[1], Paula C. F. Cardoso[2]
Juliana Galvani Greghi[1], Joaquim Quinteiro Uchôa[1], Renato Ramos da Silva[1]

[1]Department of Applied Computing/Institute of Exact and Technological Sciences
Federal University of Lavras, Lavras, Brazil

[2]Faculty of Computing/Institute of Exact and Natural Sciences
Federal University of Pará (UFPA) – Belém, PA – Brazil

{marluce,juliana,joukim,renato.ramos}@ufla.br, pcardoso@ufpa.br

Abstract. *The process of learning algorithms and programming is often a challenge for many students in courses for undergraduate in Computer Science and Information Systems. Aiming to increase learning rates, traditional teaching is giving way to new methodologies, known as active methodologies, such as problem-based learning, flipped classroom, and gamification. This paper presents an experience with a methodology that integrates all these active methodologies for teaching introductory programming to freshman students in Computer Science and Information Systems. The results indicate an improvement in learning, as evidenced by a higher number of students passing compared to those taught with traditional methods.*

1. Introduction

In the context and structure of a computing curriculum, particularly in STEM degrees (Science, Technology, Engineering, and Mathematics), it is necessary to include an introductory programming course. This course, tailored for novice students, encompasses problem-solving skills, fundamental programming concepts, the syntax and semantics of a programming language, and its practical application in developing solutions. Despite the progress in teaching methods and educational tools for introductory programming, dropout and failure rates persistently plague such courses. There remains a lack of consensus regarding the primary challenges, as well as a clear and exhaustive categorization of these obstacles (Medeiros, Ramalho, & Falcão, 2018)

According to Barcelos, Tarouco, and Bercht (2009), students often struggle with organizing their reasoning, developing problem-solving strategies, maintaining attention, concentrating, and stimulating mental calculation processes. Consequently, the skills associated with logical reasoning—such as experimenting, observing, hypothesizing, and deducing—play a crucial role in learning in this area of knowledge and can significantly impact the effectiveness of their learning process.

Morais, Neto, and Osório (2020) carried out a systematic review of the literature and presented the difficulties highlighted by works from several countries. The authors pointed out that there is a deficiency in students' competence, mainly in problem-solving, logical reasoning, and basic mathematics. Students often understand solutions presented by the teacher but struggle to reproduce them independently. There is a need for flexible study time to solve exercises, consolidate knowledge, and maintain motivation to seek solutions. Additionally, large classes prevent more individualized monitoring and the implementation of more interesting tasks. It was also noted that potential causes for students' difficulties may arise from the nature of programming, the students' background and history, attitudes towards studying, and the pedagogical strategies commonly used in algorithms and programming disciplines. These difficulties can cause failures in the disciplines of algorithms and programming until dropouts from the course.

Santos, Gorgônio, Lucena, and Gorgônio (2015) discuss that students' lack of interest in the content can be more pronounced when they are repeating a course and reviewing concepts covered in a previous semester. To address this, including complementary practical activities at the end of some classes—such as using visual and platform programming resources like Arduino—can be crucial for enhancing the teaching and learning process by boosting student motivation.

In this context, the traditional approach to teaching algorithms contrasts with active learning methods. Traditional approaches to teaching algorithms often rely on lectures, textbook readings, and assessment through written exams, which can lead to passive learning where students receive information without actively engaging with it. Active Learning (AL), in this turn, is an educational approach in which students are engaged in a controlled environment, with previous preparation, can actively think and reflect throughout this process. Three common approaches of AL are problem-based learning (PBL), flipped classroom (FC), and gamification (GAM) (Pirker, Riffnaller-Schiefer, & Gütl, 2014). These approaches help to develop critical thinking and problem-solving skills, making the learning experience more dynamic and effective. By actively involving students in their education, these methods aim to increase motivation, enhance comprehension, and reduce dropout rates, addressing many of the challenges identified in traditional algorithm instruction (Grotta & Prado, 2018; de Moraes, da Costa, & Scholz, 2022; de Oliveira, da Silva, & Rodrigues, 2022).

Thus, the objective of this paper is to describe an experience report of a hybrid teaching methodology that combines several active learning strategies in the introductory algorithms course for undergraduate programs in Computer Science and Information Systems. Our approach combines PBL, FC, and gamification for the introductory programming course using the C++ language. This article is organized as follows: Section 2 has a theoretical background of teaching methodology. Section 3 explains the methodology proposed. Section 4 presents the initial results. Finally, Section 5 presents final remarks and establishes future work.

2. Theoretical background

2.1. Traditional teaching and blended learning

High rates of dropout, failure, and withdrawal have been affecting computer programming education in undergraduate courses in Computer Science and Information Systems, both in Brazil and abroad (Zanetti & Oliveira, 2015; de Moraes et al., 2022). There are several factors that make learning computer programming a challenge, and traditional teaching approaches have not been able to address this effectively. Traditional computer programming education typically follows a structured and linear format, which includes the following main characteristics: predominantly lecture-based classes led by the instructor;

laboratory sessions to provide students with programming practice opportunities; use of a standard textbook that students are expected to follow; and assessment through written exams (Grotta & Prado, 2018).

However, especially during social isolation due to the COVID-19 pandemic, educational institutions had to explore other teaching approaches and resources, and all activities had to be carried out remotely, as noted in the works (Silveira, Bertolini, Parreira, da Cunha, & Bigolin, 2021; Dias, 2022). The authors pointed out that scenario brought many practical and technical challenges to be implemented inclusively. From an educational perspective, there was a need for the rapid production of teaching materials and monitoring tools for educational activities, incorporating Information and Communication Technology (ICT). Remote and blended learning have become more common, with greater use of online tools, learning platforms and digital resources. Furthermore, there was a greater emphasis on the flexibility and accessibility of teaching, as well as adapting methodologies to better meet the needs of students in a virtual environment.

Based on the lessons learned during the pandemic, remote and blended learning have become more common, with greater use of online tools, learning platforms, and digital resources (de Oliveira et al., 2022). Furthermore, there is a greater emphasis on the flexibility and accessibility of teaching, as well as adapting methodologies to better meet the needs of students in a virtual environment. Preuss and de Lima (2023) describe a list of tools that can help to overcome the adversities faced in introductory programming courses in the context of remote learning. This increases the debate in the field of education about the opportunities and challenges of AL in introductory programming courses. Unlike the traditional approach, AL aims for the effective participation of students and their engagement through various student-centered methodologies, strategies, approaches, and pedagogical techniques.

2.2. Problem-based learning

Problem-Based Learning (PBL) is an instructional methodology centered around students, where they collaborate in small groups to tackle real-life problems, thereby introducing new learning material. The learning process is predominantly self-directed: through problem-solving, answering questions, by discussion and analysis, students uncover theory via independent reading and study. Evaluation includes self, peer, and teacher assessment, fostering student accountability (Caceffo, Gama, & Azevedo, 2018; Goletti, Mens, & Hermans, 2021).

Yew and Goh (2016) provide evidence that the superior performance of students learning in PBL conditions, as opposed to lecture conditions, has been widely adopted in diverse fields and educational contexts to promote critical thinking and problem-solving in authentic learning situations.

2.3. Flipped classroom

In the flipped classroom model, the teacher aids students instead of merely delivering information, while students take responsibility for their own learning process and manage their own pace of study. A significant portion of the student learning process is improved by technologies, which is undertaken before classroom time. Consequently, the teacher is now able to engage with students through various learning activities such as discussions, problem-solving proposed by students, hands-on activities, and guidance (Akçayır

& Akçayır, 2018; Onyema et al., 2021). Akçayır and Akçayır (2018) also stated that, generally, the flipped model in education yields positive academic outcomes. The majority of reviewed studies reported that the flipped model promotes improvements in student learning performance. However, there is insufficient evidence to warrant generalization.

For technology courses, the use of the flipped classroom model has been shown to enhance students' achievement, self-efficacy, and interaction. It promotes active participation in the exchange of ideas and information, which can potentially increase their interest and foster critical thinking and a deeper understanding of the learning materials. (Al-Samarraie, Shamsuddin, & Alzahrani, 2020) .

2.4. Gamification

The literature summarizes "Gamification" as "the use of elements design characteristic for games in non-game contexts". Dicheva, Dichev, Agre, and Angelova (2015), in their turn, claim the penetration of the gamification trend in educational settings seems to be still climbing up to the top. But the majority of nowadays work describe only some game mechanisms and dynamics and re-iterate their possible use in educational context, while true empirical research on the effectiveness of incorporating game elements in learning environments is still scarce.

3. Our approach

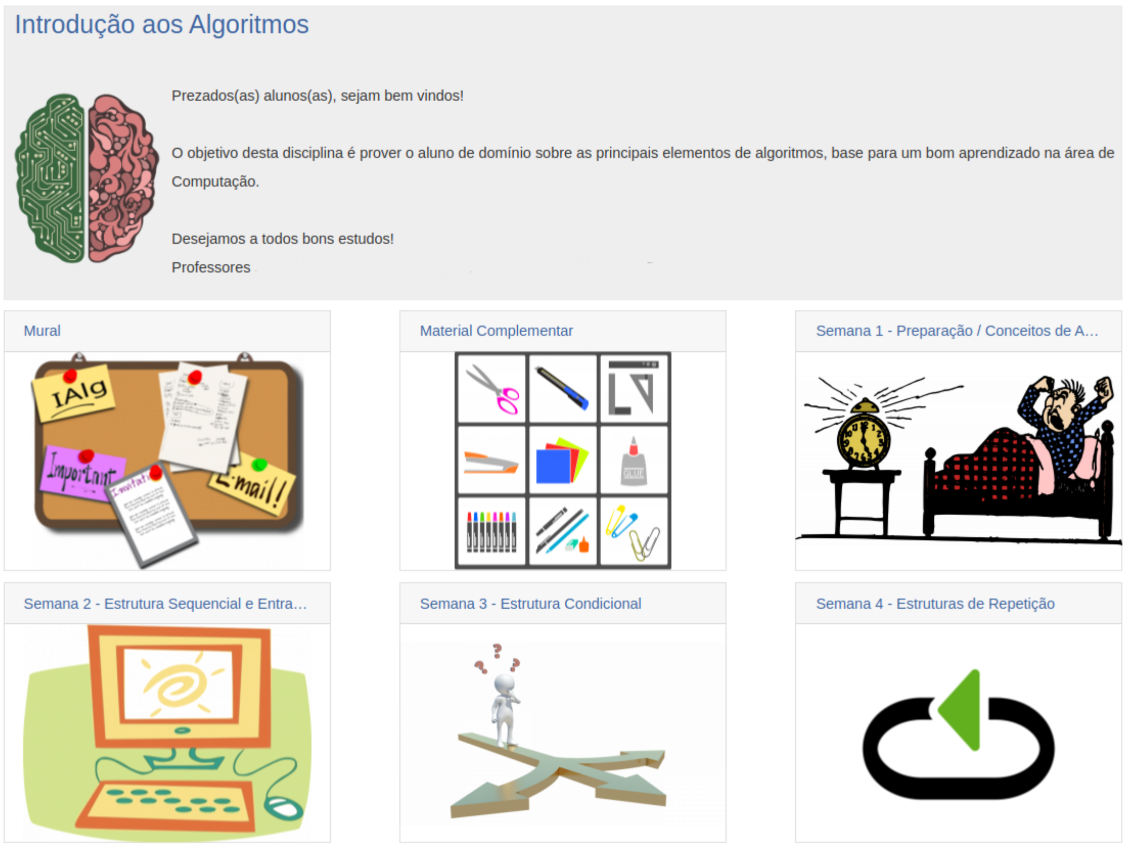
The main idea behind our approach was to combine all the positive aspects of AL into a hybrid methodology for the Introduction to Algorithms course in Computer Science and Information Systems, implemented over the two semesters of 2023 at the Federal University of Lavras. However, before delving into the methodology and materials developed by the teachers, some important information about this course is necessary. At the Institution, the course comprises two hours of remote instruction and four hours of hands-on classroom sessions, split into two sessions of equal duration. All practical classroom sessions have happened on Tuesday and Thursday. This arrangement, agreed upon with all course coordinators, where this course is mandatory, kept all classes on the same schedule and avoid disruptions due to holidays. The semester lasts seventeen weeks.

By the end of the course, it is expected that the student will have acquired the following skills: representing various everyday problems through programming logic; mastering fundamental concepts of programming logic and programming languages for the construction of simple algorithms; recognizing and mastering the basic elements for algorithm development (logical-mathematical operations and assignment; sequential, conditional, and repetitive structures; modularization and recursion; arrays, strings, matrices, and records); and properly handling text and binary files, pointers, and dynamic memory allocation. The programming language used in the course is C++.

Until the second semester of 2022, this course used a traditional teaching methodology spanning 17 weeks, with 4 hours per week of theoretical classes in the classroom and 2 hours of classes in the laboratory. During the COVID-19 pandemic, classes were conducted online with materials available in a Learning Management System (LMS) and exercises in an Online Judge Tool (OJT). An additional information, our Institution has three types to classify a undergraduate student which fails: i) when the student does assimilate and can not express it at exams; ii) by not attending a minimal classroom and iii) giving up the classroom in the respective semester or the course.

3.1. Materials

The course materials are collaboratively produced by the involved teachers. Two systems are used: a LMS and an OJT developed by the institution where the course is taught. The LMS is a customized version of Moodle (Modular Object Oriented Dynamic Learning Environment). The teachers use the LMS to provide videos, slides derived from the course's basic and supplementary bibliography, lessons with evaluative questions on weekly content, forums for questions and announcements, and additional materials and tools. Figure 1 shows the main page of the LMS where students can interact[1].


Introdução aos Algoritmos

Prezados(as) alunos(as), sejam bem vindos!

O objetivo desta disciplina é prover o aluno de domínio sobre as principais elementos de algoritmos, base para um bom aprendizado na área de Computação.

Desejamos a todos bons estudos!
Professores

Mural
Material Complementar
Semana 1 - Preparação / Conceitos de A...
Semana 2 - Estrutura Sequencial e Entra...
Semana 3 - Estrutura Condicional
Semana 4 - Estruturas de Repetição
IAlg
Important
E-mail!

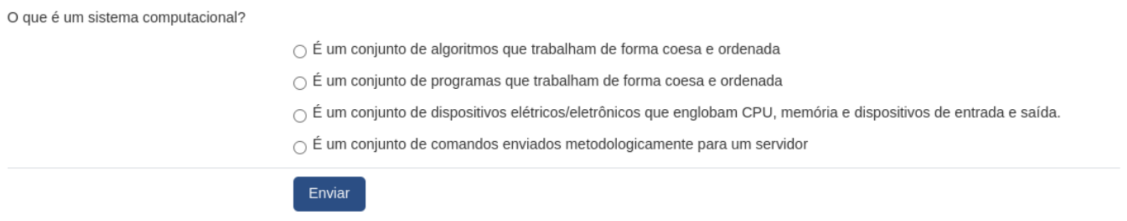
Figure 1. The virtual classroom in the LMS

The course has all classes recorded on video and made available on YouTube[2]. There are videos where a teacher explains the concepts and others where exercises are made to give an idea for the students how to solve similar problems. The slides are simple, but some changes from the previous one have increased the readability. They are also used in the recorded videos.

The lessons consist of a set of pages, each ending with questions that can be multiple choice, true or false, matching, essay, numerical, or short answer, to assess the studied content. Figure 2 shows an example of a question for the first week.

The Complementary materials and tools section contains information about IDEs, links to the OJT and the YouTube channel, instructions on how to compile using the IDE

[1]The figures are in Portuguese because it is the language used in the course.
[2]Available at `https://youtu.be/xbZL1lyc8r8?si=Gt1kmGV-E3g_9Cmw`


O que é um sistema computacional?

- ○ É um conjunto de algoritmos que trabalham de forma coesa e ordenada
- ○ É um conjunto de programas que trabalham de forma coesa e ordenada
- ○ É um conjunto de dispositivos elétricos/eletrônicos que englobam CPU, memória e dispositivos de entrada e saída.
- ○ É um conjunto de comandos enviados metodologicamente para um servidor

Figure 2. Example of Lesson

and command line, among other resources.

The OJT[3] is an online system that supports the teaching and learning process of programming, developed in the Department of Applied Computing at UFLA. It offers automatic grading support for programs written in C, C++, and Python. The system employs various types of program graders to evaluate the solutions submitted by students, assigning a score from 0 to 100 and indicating how correct is the submitted program.

The Figure 3 presents a problem example for the student solve in OJT. The structure are similar of registered problems in the system. It includes a statement explaining the problem, examples of input and output, and a section at the bottom where the student can submit their source file. In the course in question, the primary grader is based on test cases, which indicates how similar the outputs generated by a student's submitted program are to the expected outputs, based on a set of test cases previously registered by the instructor (Figure 4). For this particular problem, it is necessary to register at least 12 test cases in the system, each corresponding to a different month of the year.

In practice, OJT is important in the teaching and learning process as it allows students a certain degree of autonomy. Through this system, students receive an initial quantitative assessment of their program's correctness, enabling them to make corrections, adjust their solutions, and experiment with new evaluations. For example, if the student did not enter all possible month names to be displayed in the standard output, the result will be incorrect and the OJT will display the message explaining that ”the amount of data written by the program is different from the amount of data expected” because the correct month is missing.

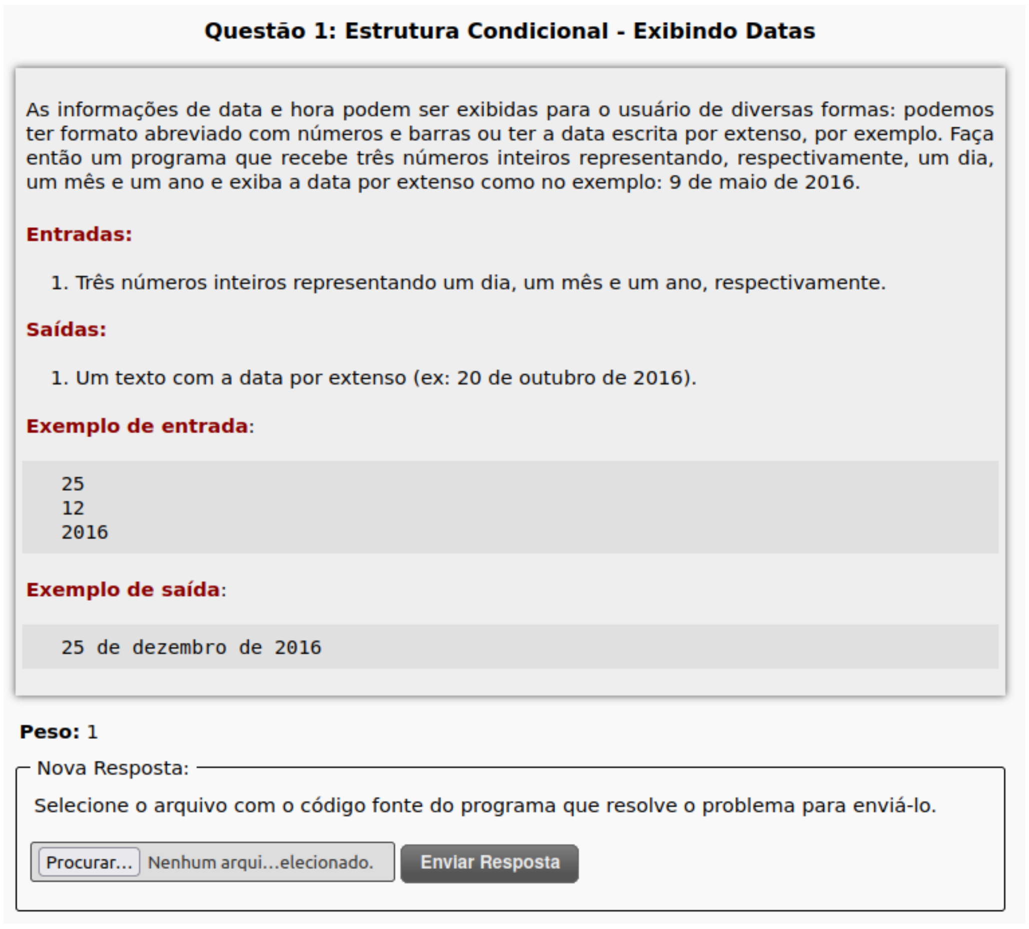
3.2. Methodology

Considering the low performance of students in the course, the experience gained with online classes, and various reports in the literature, a new methodology was developed. This methodology includes 4 hours of in-person classes and 2 hours of non-in-person (online) classes per week. The online hours are mandatory for all students, while the practical sessions are divided into groups based on the course, with a maximum of 25 students per group, with most of them containing 20 students.

The non-face-to-face (NFF) part of the course takes place in the virtual classroom of the LMS. Students are expected to develop the ability to organize and study independently, bringing their questions to the practical classes. In this approach, students will be the protagonists of their learning, while the teacher will act as a mediator.

The slides and video lessons were produced to cover the content of one week.

[3]`https://dredd.dac.ufla.br`


Questão 1: Estrutura Condicional - Exibindo Datas

As informações de data e hora podem ser exibidas para o usuário de diversas formas: podemos ter formato abreviado com números e barras ou ter a data escrita por extenso, por exemplo. Faça então um programa que recebe três números inteiros representando, respectivamente, um dia, um mês e um ano e exiba a data por extenso como no exemplo: 9 de maio de 2016.

Entradas:

1. Três números inteiros representando um dia, um mês e um ano, respectivamente.

Saídas:

1. Um texto com a data por extenso (ex: 20 de outubro de 2016).

Exemplo de entrada:

```
25
12
2016
```

Exemplo de saída:

```
25 de dezembro de 2016
```

Peso: 1

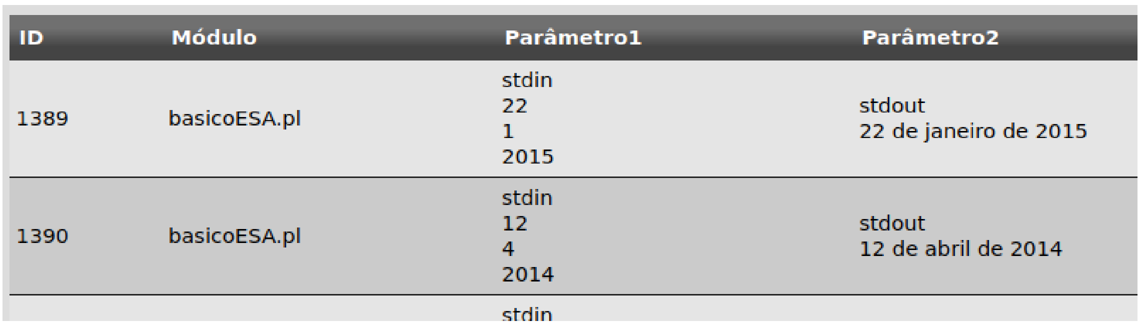
Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Figure 3. Question example in Online Judge Tool system

Yu and Gao (2022) show that videos with a duration of less than 10 minutes are more frequently watched in a flipped classroom setting. Therefore, the produced video lessons are kept short, not exceeding this limit to cover part of the content. There may be more than one video lesson for each week. The lessons on the LMS consist of a set of pages to assess the studied content. It is possible to configure navigation between the pages depending on the student's chosen response, allowing them to advance to the next page, stay on the same page, or return to the previous page.

The onsite part of the course takes place in the computer laboratory, with one student per machine. The goal during this time is for students to solidify their knowledge and address any doubts. On Tuesdays, gamified activities are conducted using tools like Kahoot (as also used in de Sousa, Maranhao, Borges, and Neto (2023)), Coding Dojo (as used by Marinho, Moreira, Coutinho, Paillard, and de Lima (2016); Scherer and Mór (2020); de Sousa et al. (2023), among others), peer review (Brown, Narasareddygari, Singh, & Walia, 2019), and individual development exercises to reinforce the week's content practically. Students are given problems to solve and can consult the teacher, course materials, and their peers. Each problem comes with a detailed description, input examples, and output examples. Each exercise emphasizes the content studied that week but may also incorporate content from previous weeks. For example, to solve a problem involving repetition structures, the solution may require reading an input file and using



ID	Módulo	Parâmetro1	Parâmetro2
1389	basicoESA.pl	stdin 22 1 2015	stdout 22 de janeiro de 2015
1390	basicoESA.pl	stdin 12 4 2014	stdout 12 de abril de 2014
		stdin	

Figure 4. Test case examples for the question in the Figure 3

conditional structures.

In Thursday's classes, the student will have to individually develop the solution to a problem and send it to the OJT that performs the correction according to the test cases registered in the system. In addition, teachers manually review the code submitted by students and provide feedback to help them understand their mistakes.

The assessment of the course includes NFF activities (lessons), participation in Tuesday's activities, completion of Thursday's activities, two practical tests, and practical teamwork based on a topic chosen by the team. This team work must meet the specified requirements to cover the entire course content.

To evaluate the students using the proposed methodology, a new distribution, from previous semesters, of points was adopted in order to evaluate group (practical project), collective (Coding Dojo, Kahoot, Peer Review) and individual participation (individual test), as presented in Table 1. The table shows two sample semesters: the first using traditional methodology and the last one employing the new approach. In this table, if the semester has more than one activity with the same name, such as 'Exam,' a number is added at the end for better understanding (e.g., Exam 1 and Exam 2).

Semester	2022/1 (traditional methodology)	2023/2 (hybrid methodology)
Evaluation Criteria	Activity List 1– 15% Exam 1 – 25% Activity List 2 – 15% Exam 2 – 25% Practical Project – 20%	Thursdays Activities (Weeks 3 to 7) – 11% Exam 1 – 20% NFF Activities (Weeks 1 to 7) – 8% Thursdays Activities (Weeks 10 to 13) – 9% Tuesdays Activities – 5% Exam 2 – 20% NFF Activities (Weeks 8 to 13) - 7% Practical Project – 20%

Table 1. Evaluation criteria in traditional and active methodology

It is important to note that there were adjustments in 2022/2, and that 2023/2 shows slight improvements and differences when compared to 2023/1. These adjustments were mostly due to the occurrence of holidays, reducing, for example, the value of Thursday activities. However, the change in assessment philosophy is noticeable when comparing these two semesters. The number of evaluative items increased and new elements began to be assessed. Instead of scoring traditional activity lists that were available for one week, for example, now some evaluated activities highlights the flipped classroom and the Tuesdays activities are performed in the classroom. With scores more effectively distributed, it is now possible to assess different aspects of learning more comprehensively, allowing for better engagement with students.

4. Results

Given that various studies have demonstrated the effectiveness of different learning methodologies in introductory programming courses, the initial goal was to propose a hybrid methodology that integrates both remote and in-person classes, along with gamified activities. Consequently, the results focus on comparing the number of student approvals under the current hybrid methodology with those from the previous traditional teaching approach.

One promising aspect was that the diverse forms of assessment encouraged students to access materials in the LMS more frequently. Yet, students posted questions in the discussion forums, answered questions about the subject, and arrived at practical classes with a better understanding of the material and specific questions. However, some students still did not develop this habit, often waiting to review or access content only during the practical classes. This indicates that there are several aspects where the strategy could be improved.

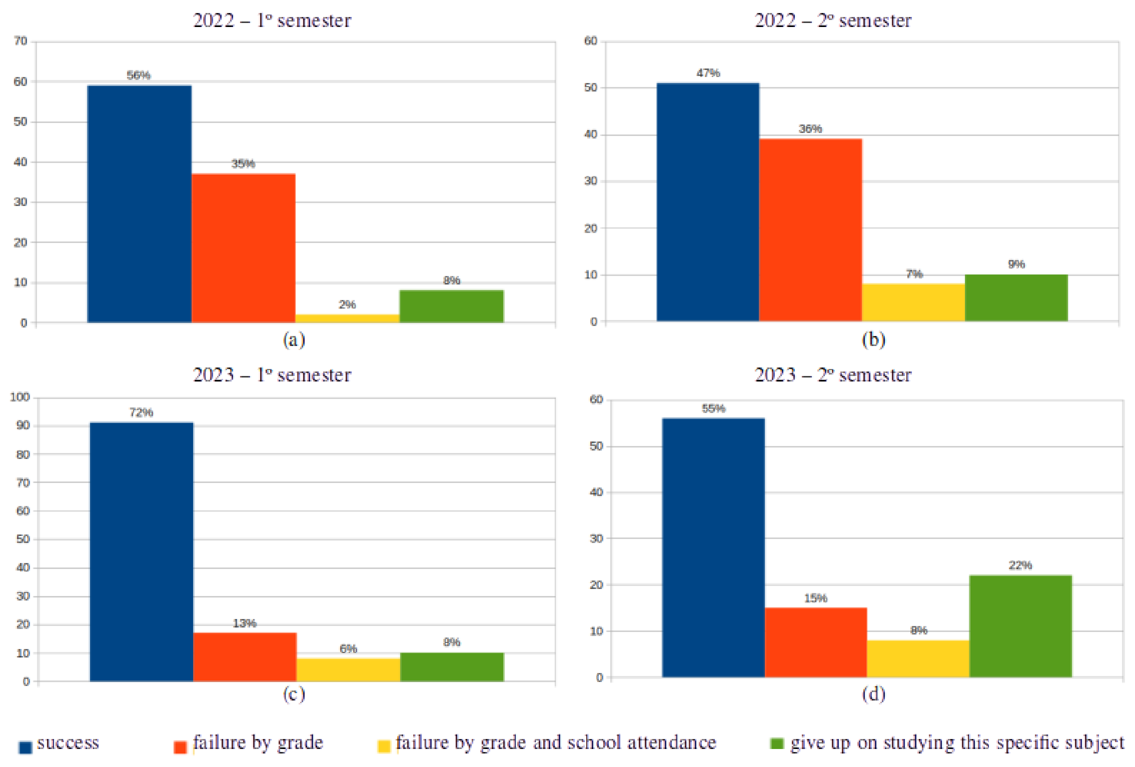


Figure 5. Results from traditional methodology and hybrid methodology

Figure 5 presents the results obtained in 2022 with the traditional teaching methodology and in 2023 with the hybrid methodology. In Figure 5a, there were 106 students, from both undergraduate programs, enrolled on the course in the first semester of 2022. Nearly half of the students failed in the course. In Figure 5b, there were 108 students, from both undergraduate programs, enrolled on the course in the second semester of 2022. Over half of the students failed, for any reason. In Figure 5c, it is shown that 126 students from both undergraduate programs were enrolled in the course during the first semester of 2023. This was the first time using the proposed methodology, and the number of failing students was significantly lower compared to previous semesters.

These results encourage us to keep the proposed methodology, with minor adjustments. The results in the second semester of 2023, presented in Figure 5d, are related to 101 students, from both undergraduate programs, enrolled on the course. There was a decrease in the number of successful students if compared to the first semester of the same year. However, the results were not worse than those obtained in 2022.

Although the number of failures have increased in the second semester of 2023, we believe that the hybrid methodology has brought benefits when considering the total number of students who passed. There are several possible reasons for this behavior, further section has a deep discussion about.

5. Discussion

By the results, it's clear that the approved number of students increases with our methodology. However, it needs to be seen with caution because this study needs more deep analysis. An important aspect to be noticed is that these students are novice, in the first semester of the undergraduate programs. Some of them are not assured to let the course to the end. Another important aspect is to know if the student evaluation format works as a bias, in favor of new methodology. Considering this as the first study is focused on presenting our ideas and sharing our good perspective.

Despite being described in the course syllabus, we noticed that on Tuesdays, students were particularly excited about gamified activities. The gamified activity, specifically using the Kahoot software, sparked enthusiasm among the students and provided tangible benefits to the learning process. The ability to provide immediate feedback and identify the class's difficulties during the game is a significant advantage, allowing for timely interventions to clarify concepts before moving forward. This not only increases student engagement but also fosters a deeper understanding of the content.

In most Coding Dojo classes, we use the kake format, where students work in self-organized pairs (a pilot and a co-pilot), using only one computer and alternating periodically (every 5 minutes). We observed that during the sessions, the co-pilot had to pay attention to their partner's coding to avoid being lost when their turn came. Students who were more adept with the discipline's concepts would explain the solution construction to their partner. We can say that the Coding Dojo facilitated learning and collaboration among the students.

The main activity behind flipped classroom was the activities at Thursdays. On those days, a list of exercises was made available and each student was randomly selected with one or two activities, most of them including implementation in the OJT. One negative result perceived by the teachers is that most students only solved the activities sorted to them, even with sufficient time to try another problems. This brings us the reflection of most students couldn't take the responsibility by their own learning, an important premise in AL. Yet, those lists were available to practice before the activity day. Unfortunately, great part of the students didn't use those resource to increase their learning.

These results have shown that the methodology is quite promising, but there are several aspects to improve and various learning questions to investigate. In a context where Information and Communication Technologies have a significant impact on education, teachers need to continuously refine their strategies to better guide students in their learning.

Final remarks

The aim of this paper was to report on a hybrid teaching methodology combined with active methodologies in teaching computer programming introduction. This methodology was applied over two periods, and although some initial positive results were observed, it is acknowledged that further investigation is required. It is emphasized that this is not a closed methodology, therefore adjustments can be made as needed, such as including other gamified activities or changing the way students are rewarded.

As future work, we plan to collect data on students' opinions regarding the teaching methodology, gather statistics on material visualization, identify learning profiles, and map periods of higher and lower student engagement throughout the learning process. As a methodological challenge, the group of teachers is interested in implementing self-paced learning, evaluating how to incorporate this approach into a relatively tied-up curricular structure.

References

Akçayır, G., & Akçayır, M. (2018). The flipped classroom: A review of its advantages and challenges. *Computers & Education*, *126*, 334–345. doi: https://doi.org/10.1016/j.compedu.2018.07.021

Al-Samarraie, H., Shamsuddin, A., & Alzahrani, A. I. (2020, 6). A flipped classroom model in higher education: a review of the evidence across disciplines. *Educational Technology Research and Development*, *68*, 1017-1051. doi: 10.1007/s11423-019-09718-8

Barcelos, R., Tarouco, L., & Bercht, M. (2009). O uso de mobile learning no ensino de algoritmos. *Revista Novas Tecnologias na Educação*, *7*(3), 327–337. doi: https://doi.org/10.22456/1679-1916.13573

Brown, T., Narasareddygari, M. R., Singh, M., & Walia, G. (2019). Using peer code review to support pedagogy in an introductory computer programming course. In *2019 IEEE Frontiers in Education Conference (fie)* (pp. 1–7). doi: 10.1109/FIE43999.2019.9028509

Caceffo, R., Gama, G., & Azevedo, R. (2018). Exploring active learning approaches to computer science classes. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 922–927). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3159450.3159585

de Moraes, R. P., da Costa, V. F., & Scholz, R. E. (2022). Mapeamento sistemático do ensino introdutório de programação nos ensinos técnico e superior no Brasil. *Revista Brasileira de Informática na Educação*, *30*, 628–647. doi: 10.5753/rbie.2022.2611

de Oliveira, M. G., da Silva, M. F., & Rodrigues, C. B. (2022). Curso híbrido baseado em moocs de lovelace e oficinas presenciais para aprendizagem ativa e nobre de pensamento computacional e programação. In *Anais do XXVIII Workshop de Informática na Escola* (pp. 179–188). doi: https://doi.org/10.5753/wie.2022.225669

de Sousa, J. S., Maranhao, D., Borges, P. V., & Neto, C. d. S. S. (2023). Uma análise da integração da metodologia ativa coding dojo a uma plataforma de ensino-aprendizagem de algoritmos. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação* (pp. 127–138). doi: https://doi.org/10.5753/sbie.2023.234703

Dias, A. I. d. A. S. (2022). Pensamento computacional para gerar soluções inclusivas no contexto universitário pós-crise da pandemia de covid-19. In *Anais do I Workshop de Pensamento Computacional e Inclusão* (pp. 63–72). doi: https://doi.org/10.5753/wpci.2022.226879

Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in education: A systematic mapping study. *Journal of educational technology & society*, *18*(3), 75–88.

Goletti, O., Mens, K., & Hermans, F. (2021). Tutors' experiences in using explicit strategies in a problem-based learning introductory programming course. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education v. 1* (pp. 157–163).

Grotta, A., & Prado, E. P. (2018). Um ensaio sobre a experiência educacional na programação de computadores: a abordagem tradicional versus a aprendizagem baseada em projetos. In *Anais do XXVI Workshop sobre Educação em Computação.*

Marinho, C., Moreira, L., Coutinho, E., Paillard, G., & de Lima, E. T. (2016). Experiências no uso da metodologia coding dojo nas disciplinas básicas de programaçao de computadores em um curso interdisciplinar do ensino superior. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação* (Vol. 5, p. 1097).

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. doi: 10.1109/TE.2018.2864133

Morais, C., Neto, F. M., & Osório, A. (2020). Difficulties and challenges in the learning process of algorithms and programming in higher education: a systematic literature review. *Research, Society And Development*, *9*(10). doi: https://doi.org/10.33448/rsd-v9i10.9287

Onyema, E. M., Choudhury, T., Sharma, A., Atonye, F. G., Phylistony, O. C., & Edeh, E. C. (2021). Effect of flipped classroom approach on academic achievement of students in computer science. In *Data driven approach towards disruptive technologies: Proceedings of midas 2020* (pp. 521–533). Singapore. doi: https://doi.org/10.1007/978-981-15-9873-9_41

Pirker, J., Riffnaller-Schiefer, M., & Gütl, C. (2014). Motivational active learning: engaging university students in computer science education. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer ScienceEeducation* (pp. 297–302).

Preuss, J. O., & de Lima, C. C. (2023). Ferramentas online na aprendizagem de programação de computadores no contexto do ensino remoto. *Revista Brasileira de Informática na Educação*, *31*(1), 790–813. doi: https://doi.org/10.5753/rbie.2023

.2867

Santos, A., Gorgônio, A., Lucena, A., & Gorgônio, F. (2015). A importância do fator motivacional no processo ensino-aprendizagem de algoritmos e lógica de programação para alunos repetentes. In *Anais do XXIII Workshop sobre Educação em Computação* (pp. 168–177).

Scherer, A. P. Z., & Mór, F. N. (2020). Uso da técnica coding dojo em aulas de programação de computadores. In *Anais do XXVIII Workshop sobre Educação em Computação* (pp. 6–10).

Silveira, S. R., Bertolini, C., Parreira, F. J., da Cunha, G. B., & Bigolin, N. M. (2021, maio). Impactos do ensino remoto na disciplina de paradigmas de programação durante o isolamento social devido à pandemia de covid-19. *Revista Gestão e Desenvolvimento*, *18*(2), 200–213. doi: 10.25112/rgd.v18i2.2455

Yew, E. H., & Goh, K. (2016). Problem-based learning: An overview of its process and impact on learning. *Health professions education*, *2*(2), 75–79. doi: https://doi.org/10.1016/j.hpe.2016.01.004

Yu, Z., & Gao, M. (2022). Effects of video length on a flipped english classroom. *Sage Open*, *12*(1), 21582440211068474. doi: 10.1177/21582440211068474

Zanetti, H., & Oliveira, C. (2015). Práticas de ensino de programação de computadores com robótica pedagógica e aplicação de pensamento computacional. In *Anais dos workshops do Congresso Brasileiro de Informática naEducação* (Vol. 4, p. 1236). doi: https://doi.org/10.5753/cbie.wcbie.2015.1236