Insights after 42 Months of Application and Development of a Computational Thinking Methodology for Children

Rafael Monclar
LUDES - PESC/COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, Brazil
rstudart@cos.ufrj.br

Geraldo Bonorino Xexéo
LUDES - PESC/COPPE e DCC/IM
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, Brazil
xexeo@cos.ufrj.br

***Abstract*—In the Brazilian scenario, 77% of public schools and computer science education are still far from the reality of developed nations. The object of this article is the evolution of a methodology in the education of computational thinking for children from 5 to 10 years old, which is the result of a work in progress started three and a half years ago. In it are formal definitions of what computational thinking is and, later on, the limitations and problems due to technical infrastructure issues, and attention problems by the selected target audience are also included. The solutions to this come from computer science unplugged and behavioral study to stimulate child engagement. These, added to an analysis of the educational scenario, form a basis for the methodology being designed throughout these past three and half years, which has in its teachers and students, as well as in the teaching environment, its main variables. The results observed so far indicate that children from 5 to 10 years of age tend to remain more engaged while studying computational thinking than older ones. In addition, activities in pairs, and even trios, with children of this age group, produced a higher level of involvement among them. It was also observed that a teacher's technical experience is not essential for the good conduct of a class, since hers, or his, interpersonal skills are what make the difference when dealing with younger children.**

***Index Terms*—education, computational thinking, computer science unplugged.**

I. Introduction

Although the term "computational thinking" was only coined in 2006 [1], the first steps towards this type of teaching come from the 1970s [2]. And, despite almost fifty years of programming teaching theories for children and adolescents, there are several obstacles such as the complexity of the domain, which includes elaborate knowledge associated with abstract concepts, in addition to the arduous development of cognitive skills of a practical nature, such as logical reasoning, problem solving and high level abstraction [3]. What differentiates the current generation from those of Papert's time is the fact that they are digital natives. Children born in the 21st century already have access to digital media, which could make the incorporation of programming logic more immediate [4].

Currently, it is possible to find analog toys and digital games [3] that playfully present computational concepts such as pattern recognition and algorithmic thinking, two of the main pillars of the area [1]. In parallel to this, there is a lot of didactic material online, and in books, for teaching computational logic to children and adolescents [5].

Considering the existing methodological vacuum and the relevance of this concept for children [2], a course with this theme started to be applied, by one of the authors of this paper, in 2016.

The desire to promote a more inclusive education focused on the children's literacy period [6] helped to shape the content of the work presented here. However, the original idea was to promote the teaching of computational thinking to children of a broader age range, ranging from 6 to 17 years of age.

This original choice had been based on the benefits of computer teaching already advocated by researchers such as Papert [2], who since the last century has been trying to include this type of discipline in K-12 classrooms, which is the range of years of supported primary and secondary education found in the United States, going from kindergarten to 12th grade.

In parallel, the popularization of computational concepts [5], the growing government incentive [7] and the emergence of tools for the production and study of computer science [3], [8], in addition to educational games with this thematic, as seen in [3], point to the scenario envisioned by Papert years ago.

However, as popular as technological devices have become in 2020, computers, smartphones and tablets are not yet a reality for a large part of the population of countries on the capitalist periphery. This is reflected not only in the access of students to such technologies, but also of teachers and educational institutions [9].

Considering that 77% of schools are public in Brazil [9], and that investment in education has been frozen for the next 20 years [10], there is no prognosis of improvement in the infrastructure of most educational institutions that contemplate the absorption of cutting-edge technologies to guarantee the type of education necessary to aggregate the competitive differential of future brazilian citizens [11].

This means that the original research that motivated the first steps of this work went through refinements and changes in

perception, including a notion of making the teaching of computational thinking more inclusive. However, it is important to present the entire trajectory that made the research reach this present moment.

The aim of this paper is to present the main insights from development and application of a methodology for teaching computational thinking to children. In particular, one that was able to make use of computer science unplugged, making this type of knowledge more accessible to low-income children who did not have access to technological devices.

Although computational thinking was the main motivator behind the course content, it was necessary to better understand the Brazilian educational scenario, computer science unplugged, and, finally, student engagement, a problem that was detected in the first months of application of the before mentioned course.

In the next sections we present the theoretical background of this research (Section 2), followed by the related work (Section 3), the methodology presented (Section 4), the history and insights gathered throughout the years of application of this teaching's type (Section 5), ending with the conclusions (Section 6).

This section describes the main areas on which this work is based, ranging from computational thinking to student engagement.

A. Computational Thinking

Since 2006, the term "Computational Thinking" has appeared in academic literature thanks to Jeannete Wing [1]. The author's defense was that such a way of thinking should be adopted by all citizens, including young people and children, as a way of developing the skills and knowledge derived from the cognitive model achieved by computer science [1], [12], [13]. Due to their transdisciplinarity and universality, such capacities could be useful to everyone and not just professionals in this field.

Later, Wing presents computational thinking as a set of thought processes equivalent to what a computer, human or machine, would produce by having to formulate a problem and express its solution effectively [14]. In other words, computational thinking focuses on the mental step-by-step until a computational solution to the previously formulated problem is reached. This solution, in itself, can be achieved both by a human being and by a machine. This means that an individual can learn computational thinking without necessarily having access to a computer device. Some researchers, however, argue that this area is still in its early stages of maturity, with no ready definition [15], [16].

According to Wing, computational thinking is not only about problem solving, but also about its formulations [14]. One of the important points highlighted by her is that computational thinking is much more than the ability to program [1], as it focuses on conceptualization and requires different levels of abstraction.

This logic is reinforced by research [17] that says that computational thinking is essentially about data and ideas, as well as their combination for problem solving. Technologies and equipment are secondary to this [17].

Multiple pillars are worked and developed with computational thinking, such as [1], [13]: decomposition of problems, abstraction, pattern recognition, algorithmic thinking, simulation, recursive thinking, parallel thinking, automation, modeling, and etc.

Other researches [18], [19] follow a similar line, indicating that computational thinking is a way of reasoning used in problem solving by computer scientists. Children can learn algorithmic thinking from an early age and start using it to solve problems for the rest of their lives [20].

Computer scientists skills such as logical reasoning, problem solving and high level abstraction, can be fundamental for scientists in other areas, as well as for ordinary people, interested in solving common problems [21], which could be an additional stimulus in times of scientific negationism.

The debate on concepts and characteristics of computational thinking also addresses the mental and cognitive abilities associated with the methodology for problem solving linked to computer science. Because of this, computer learning is characterized, but not limited, to the following aspects [20]:

- Problems formulation in order to allow the use of a computer, or other tools, to solve it,
- Data organization and analysis in a logical way,
- Data representation through abstractions, such as models and simulations,
- Automation of solutions through algorithmic thinking,
- Identification, analysis and implementation of possible solutions in order to find the most efficient combination of steps and resources, and
- Generalization and transfer of this problem solving process to a wide variety of other types of problems.

This is based on a set of attitudes and predispositions, which are essential dimensions of computational thinking, as [20]:

- Confidence in dealing with complexity,
- Persistence in working with difficult problems,
- Ambiguity tolerance,
- The ability to deal with open and finite problems, and
- The ability to communicate and work with others to achieve a common goal or solution.

The ability to solve problems is also a common feature in this effort to collectively build a frame of reference for computational thinking for education [21].

The perspective adopted by the National Science Foundation (NSF) is based on the seven "big ideas" of computing, [22]:

- Computing is a creative human activity,
- Abstraction reduces information and details to focus on concepts relevant to understanding and solving problems,
- The data and information serve to facilitate the creation of knowledge,
- Algorithms are tools to develop and express solutions to computational problems,
- Programming is a creative process that produces computational artifacts,

- The digital devices, systems and the networks that interconnect them allow to foster computational approaches to solve problems, and
- Computing allows innovation in other areas, including the social sciences, human sciences, arts, medicine, engineering and business.

As can be seen, computational thinking, despite its relevance, does not depend exclusively on a technology to be taught. Thus, we arrive at the concept of our next item of our theoretical foundation.

B. Computer Science Unplugged

Computer science unplugged, or unplugged computing, is a set of activities and ideas widely used in order to engage diverse audiences in subjects related to computer science, without those involved needing to learn to program or even use a digital device [23].

The idea originated with a focus on students from 4 to 11 years old, in order to help them understand what computer science could involve besides programming [23].

The activities of this method, in addition to being kinesthetic, tend to be accessible, providing greater student engagement without a computer or typing skills [24]. It is even frequently mentioned in books on teaching computer science [25], [26], as well as being used as a pedagogical technique on "coding" sites, such as *Code.org*[1].

The unplugged computing approach is found in curriculum recommendations and projects, including: activities suggested by ACM's K-12 grid in 2003 [27]; as part of the design of an elementary school curriculum [28]; as part of the course "Exploring Computer Science" [29]; and as a support for the Australian Digital Technologies curriculum [30].

The term came from a collection of activities shared digitally in various formats in the early 1990s, culminating in a free digital book in 1999 [31]. Originally, the material could be adapted for use in classrooms, but to do so, it was necessary to consider how to link activities to curriculum objectives and broader computing issues, in addition to assessing student progress [32]. Only recently has the material been organized into lesson plans for use in school curricula, including links to plugging it in activities, which are ideas for programming-related exercises. Unlike the other versions, the most recent one was published as a site and not as a book, although all versions mentioned here are available on the main site [33].

The official website organizes a series of principles on which it is based, including: fun, learning by doing, the absence of the need for computers, and resilient activities. The so-called unplugged material is not intended to be a curriculum in itself, or a program of study. Its objective is to be a pedagogical method with several possible benefits, such as:

- the barrier of programming learning, which can be seen as an insurmountable obstacle for some, is removed as a prerequisite for learning about important premises of computer science [34],
- students can be involved in a meaningful way with the broader and more perennial issues addressed by computer science [35], avoiding the common mistake that the focus is only on programming [36]. This approach supports a spiraling curriculum, where students are able to maintain an overview of where their learning is going, instead of getting bogged down in the details of just one aspect of the subject [37],
- it can be used in situations where there are no computers available or, if there are, they end up being a distraction for students, in addition to other technical problems such as having to install certain software for the class in question, for example [38], and
- if there is little time available, or a very large audience, such as, for instance, a 15-minute publicity, or a presentation at a science fair. In such cases, it is easier to engage students with unplugged computing exercises than to try to get them to program something [39].

New activities were developed by other educators, many of them shared on the *CS Unplugged* website, but there are many others created independently, including books by other authors with the theme unplugged [40], [41].

This means that the unplugged computing approach has become more than a specific set of activities, which, in turn, is evolving. Some guidelines on how such activities can be designed are provided [42], however, the key principles identified and which support the unplugged computing approach are:

- avoid using computers and programming,
- has the essence of play or challenge for the student to explore,
- be highly kinesthetic,
- take a constructivist approach,
- contains short and simple explanations, and
- a sense of history.

Because of the variety of materials emerging, the term "unplugged" came to be used for computer science teaching activities that do not involve programming and, therefore, the term came to refer to a general pedagogical approach. Due to the subtle differences between each of these forms of "unplugged", it is important to be aware of what context is being referred to when a single activity or event is assessed [37].

The main disadvantages found in this type of teaching come from the fact that unplugged computing needs careful preparation. Most of the time, it gives teachers a lot of work. Not only in the preparatory processes, but even because some see such activities as barriers. This is due to the need for a change of perspective between something strictly related to computer science to broader situations, which makes some teachers need to change their teaching style. In addition, it is difficult to keep track of time and stay focused throughout the activity of the genre. Finally, it can happen that students

[1] http://www.code.org

simply refuse to participate, since "playing" cannot always be seen as something engaging [43], [44].

C. Children's Educational Engagement

Research on classroom engagement shows psychological and behavioral characteristics [45]–[47]. Psychologically, engaged students are intrinsically motivated by curiosity, interest and pleasure, and are likely to want to achieve their own intellectual or personal goals. In addition, the engaged child demonstrates concentration, enthusiasm and effort behaviors [48].

There is a difference between being engaged and doing a task. Engagement is something active [49]. It requires students to be attentive and present; it requires that there is a commitment to the task on the part of the student and that he seeks some inherent value in what was asked to be done by him. The engaged student not only performs the assigned task, but also does it with enthusiasm and diligence. In addition, the task is performed by the student because he realizes that it is associated with a short-term goal that he values.

According to research, as children progress from primary school, to middle school, to high school, disengagement only increases [46], [50], [51]. They may lose interest in classroom activities, respond poorly to teacher guidance and classroom interaction, and do significantly worse in assessments. Studies have shown that such patterns of educational disengagement start from the third grade [52].

As important as engagement is for the success of children as students, strategies to promote it are not promoted, or even present in the vast majority of school environments [47], [53]. Lessons that promote passivity, mechanical learning and routines tend to be the rule and not the exception [54], [55]. And low levels of engagement in children increase the risk of disruptive behavior, absenteeism and, eventually, school dropout [56].

The use of engagement strategies serves as a teaching tool because [48]:

- focuses on children learning,
- supports the learning of specific skills and concepts, and
- provides children with positive associations with learning.

Engagement strategies can be used for different purposes and in different situations. However, the characteristics that engaging experiences need to have are [48]:

- the activation of prior knowledge,
- the incentive to active investigation,
- the stimulus of collective interaction,
- encouraging collaboration,
- freedom of choice,
- the inclusion of games and humor,
- support for proficiency,
- stimulating independent thinking, and
- does not make children wait.

There are several engagement strategies that follow these characteristics and work in both large and small groups, including private lessons. However, only KWL will be mentioned here, so that an example could be given [48].

The KWL should be used whenever a new study or topic is presented to children. In this case, they should be asked "What do you already *know*, what do you *wonder* about, and what do you want to *learn*?". In this case, this strategy indicates to children that their prior knowledge and interests are taken into account [57].

II. Related Work

Seeking a greater understanding of the area of teaching programming and computational thinking, a search for related work was carried out.

The work [3] first presents the problem of teaching programming and proposes using games with purpose as a solution. A meta-analysis of research and development on these types of games focused on teaching programming is done. Although the article had higher education as its initial target scope, its analysis ended up identifying that 50% of the games found to teach this type of content were focused on children.

In [3], evidence was found that games with the purpose of teaching programming developed with current processes may not contribute to the development of students' logical, cognitive and social thinking [58]. This happens mainly because the teaching tools found, by themselves, are not able to engage students for a long period.

Some papers indicate the value of knowing computer science as a way to learn other disciplines, even those that are not part of STEM (science, technology, engineering and mathematics). One of them even implements a six-week intervention at a British primary school with the aim of assisting literacy through computing and storytelling [11].

Game development appears in a recurring way as a method of trying to achieve computational thinking in works such as [59] and [60]. In fact, in [61] a list is presented of what must be done to promote this form of teaching in US public schools. However, the target audience was children over 10 years old.

Two papers, specifically, stood out for directly addressing the main subject in this research, as they aim at creating computational thinking teaching methods.

MECOPROG is the proposal of [62]. In it, 132 students from Spain, between 9 and 12 years old, were evaluated before and after the application of the methodology, which is based on analogies to explain concepts of computational thinking together with the Scratch[2] tool. As examples, a mixer was used to explain the loop block and a smart refrigerator to talk about conditionals. Although the preliminary idea of such a methodology is valid, some problems with the work in question have been identified. One of them concerns the MECOPROG assessment itself, since it was done with children who were already studying programming in their schools. In addition, children improvements regarding content acquisition, although significant in their 15%, could not be checked against a control group. Finally, there is still the reliance on the use of programming as a way of teaching computational thinking.

[2]http://scratch.mit.edu

The second work does not produce a methodology per se, but suggests alternatives for this. The process starts by looking for a way to express the idea you want to convey, be it writing, drawing, using Lego, or even theater. Then, a possibility is presented to add creative methods in computer science tasks, and vice versa [63]. Finally, according to the authors, it is said how to insert creative activities of unplugged computing in the teaching process. However, the biggest flaw of the job lies in the fact that nothing genuinely new is shown, mainly because such a method of choosing activities and creating them ends without being tested in practice.

III. Methodology

A methodology is nothing more than the direction to achieve some objective [64]. And the methodology that has been developed in this work based on several iterations of the course originally thought in 2016, currently aims at teaching computational thinking.

Several transformations in this methodology have been made based on observations during its applications. One of them, for example, was the change of the target audience, which currently is specifically children from 5 to 10 years old.

As a base methodology, constructivism was chosen as there is a growing demand for greater interactivity to be incorporated into learning materials [65]. The constructivist method of design is different from the task-oriented linear method of an instructional system design approach. Designers who use a constructivist method to create learning environments are less focused on a how-to or process approach, but emphasize the elements that facilitate a learning process [66].

Classes are planned and prepared in advance, always with a digital presentation, as showed in Figure 1. However, it is only used as support for an initial presentation of about 5 minutes on the content that will be presented in that class. That is, the teacher should be able to present the same content using a whiteboard or blank paper sheets and colored pens.

Regarding class time, it should be 30 to 40 minutes for children aged 5 to 7 years and 60 minutes for children aged 8 to 10 years.

For children from 5 to 7 years old, digital creativity and computational thinking should be taught using games with purpose [3], Scratch Jr. and unplugged computing activities. For older children, 8 to 10 years old, game creation and programming using Scratch should be the choice for teaching.

Activities without technological devices and unplugged computing must be prioritized among the youngest, and the additional material for carrying them out must be prepared in advance by the teacher and provided to the students.

Teaching needs to be socio-individualized, meaning that it must value attending to individual differences, but taking into account the socialization among students [67]. Thus, classes should not have more than 10 students, ideally 6 to 8, especially in classes with children up to 7 years old. In groups of students aged 8 to 10 years, double programming activities are encouraged.

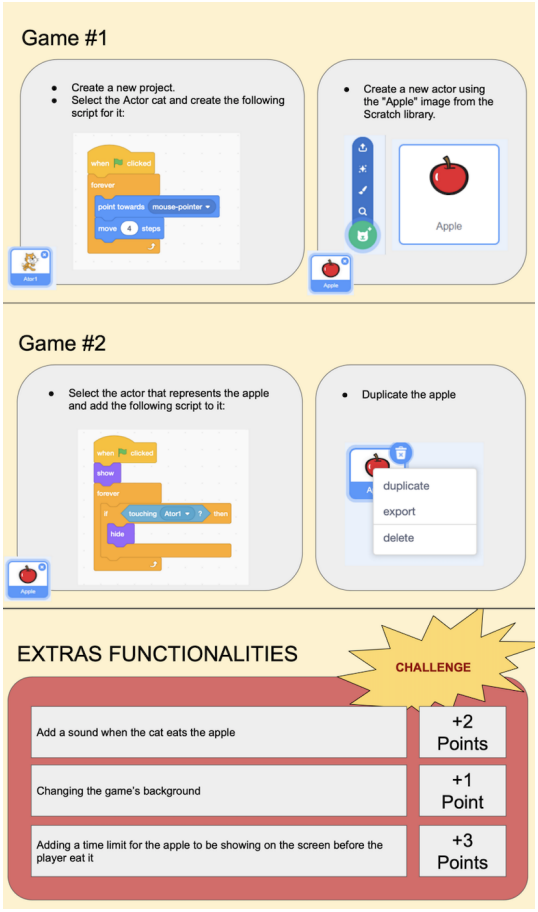

Fig. 1. Example of a presentation from one of our first plugged activities. It teaches concepts such as loops, directions and movements in the Scratch tool

There should be a focus and concern with engagement [48] whenever presenting a new activity or concept, so it is important to keep in mind the KWL (Know, Wonder and Learn) approach.

The methodology is based on activities for students aged 5 to 7 years and projects for those aged 8 to 10 years. The main difference between the two is the time to carry out and the depth of the concepts explored. Activities have a beginning, middle and end in the same class and should be as close as possible to playing. The projects can last up to two classes and, although fun, have a higher degree of difficulty.

All projects need to present two levels of objectives in their achievements: the basic and the extra challenges. This should be created to keep the engagement of children who are able to resolve basic goals more quickly.

Finally, there should be no homework, but a suggestion for an additional task. Also, there should not be exams. This must be done so that the teaching of this content is not associated with something mandatory. However, the evaluation of the student's performance can be verified by the teacher because of the tasks presented in each class, mainly for the classes are small and it is possible to personalize the teaching in this way.

IV. Discussion

The experience of teaching undergraduate students, however great, is put in check when the target audience changes for children as young as five years old. Nevertheless, before presenting the final idea of the methodology, it is necessary to outline a timeline of about forty-two months containing the observations and analyzes arising from the application of a course to teach computational thinking to children.

A. Current Education Scenario

In many urban areas, the predominant model of public education - the typical and comprehensive college of the 1950s and 1960s - and, most importantly, the curriculum and pedagogy of these schools are out of date and do not allow all students to reach high academic standards or develop skills ranging from civic skills of tolerance, intergroup communication and conflict resolution to participation in public life. Many students in these schools are disengaged and disconnected from each other, yet digitally connected, and they see no purpose in their education. There are no opportunities for these students to take responsibility for their own education. They are adrift, passing schools with poorly trained teachers, in large classrooms with little participation, practical experiences or encouragement for critical reflection. These students exist as anonymous among their teachers and many of their peers. These schools are poorly equipped to address the main academic challenges facing urban students: social and development issues, the increasing complexity of the material they are expected to learn, in addition to their own alienation from the school [68].

This is a general scenario presented by those responsible for "Quest to Learn", a model school, free for its students, and that seeks to develop its teaching structure based on gamification, passing through all layers of Bloom taxonomy [69]. It is a reality, particularly, found among students over 10 years old and in developed nations. However, this type of geopolitical and social approach serves as a basis for understanding some of the several limitations that we encounter when trying to teach any discipline, and, in particular, one that depends on technological resources in public schools in underdeveloped or developing nations.

Almost a third of all students drop out of school in the United States, and only 50% of Latin, African American and Native American students have completed high school there [70]. Of these, only 35% claimed to have done so because they were "failing at school". Almost half, about 47%, said that "the classes were not interesting" [68]. These are data from 10 years ago, a time when smartphones were starting to become popular and 4G started its commercial operations.

Although many students are alienated from the school, other data show that their use of digital media only increases. In March 2005, the Kaiser Family Foundation released a report that found that, on average, young men and women aged between 8 and 18 are exposed to 8 hours and 33 minutes of digital and other media (defined as the Internet, music, video games, television and movies) daily. In the same year, a study by the Pew Internet and American Life project reported that 57%, or about 12 million, of online teenagers between the ages of 12 and 17 are content creators like blogs; personal web pages; pages for a school, friend or organization; original works of art, photos, stories or videos; and remixed content that produces a new creation [68], [71].

The appeal to digital media only grows, making it pervasive, but still unattainable in most educational institutions. While significant gains have been made in providing minimal access to a computer and the Internet for most young people in American schools and libraries, the latest technologies continue to evolve at a faster rate than these institutions can keep up with. Low-income communities lag far behind in purchasing computers and providing high-speed connections. In addition, access to technology has become less important than developing the skills and content needed to participate in this rapidly evolving technological trend [72]. In the past, access to books, visits to museums and shows were used to draw the line between the social practices of low and middle income communities. However, access to technologies and their related social experiences is playing a similar role in today's society [68].

1) The Brazilian Case: Brazil is currently experiencing a crucial period in its history, marked by multiple crises and a strong disbelief of the population in the capacity of the public power to manage to reverse serious political, economic and social problems that affect, to a greater or lesser extent, life of all Brazilians. Faced with countless obstacles that plague the country, the urgency of some central changes has gained prominence in the public debate. Among them, the need to decisively improve the quality of basic education. After all, even if education does not solve all of Brazil's challenges, research and worldwide experience already show that without it, there will be no lasting economic recovery or significant advances in the current situation. Even more considering that there are 181 thousand schools in Brazil, 77.7% of which belong to the public network [9].

The Basic Education Development Index (Ideb) of 2018, released by the Ministry of Education and the National Institute of Educational Studies and Research Anísio Teixeira (Inep), shows that Brazil did not reach the goals in the final years of elementary and high school . However, the results with students from the early years (1º to 5º year) of elementary school were the only ones in which the goals were above planned, reaching 5.8. No state reached the target in high school, whose expected was 4.7 but it only reached 3.8 [73].

These data suggest that something happens after students are 11 years old, causing their performance to drop. However, no research has been found that accurately indicates the cause. However, what this consolidates is a scenario of possibilities that are better used by children under 11 years of age, in comparison with students above that age.

One study, which heard four thousand teachers from Elementary School (34%), Middle School (34%), and High School (27%) and from Youth and Adult Education (5%) from public schools across the country, in the first half of 2017,

points out that more than half (55%) of Brazilian public school teachers use digital technology regularly in the classroom, and that the most frequent limiting aspects for the use of technological resources are the lack of infrastructure - as little equipment (66%) and insufficient internet speed (64%) - and the lack of adequate training - 62% have never taken general courses in information technology or digital technologies in education [74].

Although the study fails to objectively clarify what is considered to be "use of technologies in the classroom", it is possible to infer based on the technical report that it would be the use of computers, tablets, or smartphones. These, in turn, with a focus on the use of word processor, spreadsheet, internet, educational technologies, educational software or games, programming courses or application development.

Still, the aforementioned problems show that there is a gap that needs to be filled, however, without predicting when this will occur. In addition, it is necessary to address the problem of children's educational engagement.

B. History and Insights

The work that motivated this research started to take place when a computer driven course initiated its activities in a private educational institution in the city of Rio de Janeiro. Originally, the idea was to bring elements of gamification to the course, since the project started with the creation and programming of games.

Gamification was considered and even implemented in the early stages of research to try to improve levels of student engagement. The works of [75]–[77] were used as the basis for this.

Over the period of three and a half years, and passing through other educational institutions, several modifications were made in this "course to teach game programming" for kids, and a real methodology started to become necessary. Ideas such as the use of gamification were abandoned, for instance, because the engagement of the students didn't corresponded to what was expected. In other words, only one student among 12, from two different classes, changed his attitudes and became more focused in comparison with the non-ludified course. In addition, research and surveys such as those of [9], [68], [71], [74], made gamification secondary in this process of implementing a computational thinking course.

The first prototype of the course, which lasted 10 weeks, had 7 students, aged between 10 and 13 years old, and used Scratch 2.0 as a tool, focusing on the development of digital games. After two and a half months, the participants' satisfaction was considered good to excellent, based on unstructured conversations with parents and students. However, when the course went on to its second module, after the end of the year holiday period, only 3 of the 7 children continued (42%). Talking to liberal professionals who apply extracurricular courses such as theater, swimming and drawing, such evasion seemed to be common, since many parents choose to enroll their children as a way to test their skills, even so that a future career choice could be made easier. The original menu of the course, created by a teacher with 10 years' experience in the game development area and 20 in the teaching profession, was carried out under the supervision of its creators. However, there was no methodology per se, except for the sequence of disciplinary content that should be delivered focusing on programming. The four modules created, therefore, went from Scratch to the use of Javascript, with PHP and MySQL, through Python, with Minecraft, and their subsequent use of the Pygame library.

The growth in the number of students in the second year of the course was substantial, reaching 60, which meant that more teachers needed to participate in the project. In this stage of the process, there was the first contact with children from 6 to 10 years old, bringing new challenges and showing that there was a gap that would need to be filled in this teaching model. It was then that the focus ceased to be programming for children and adolescents by itself, to become a school of computational thinking that used programming to achieve this goal. However, deficiencies in the method were already beginning to be noticed, which did not work in any specific way on students' engagement and sense of purpose.

At the end of this second year, however, an anonymous evaluation form using the Likert scale was made available to parents and students aged 10 and over. Of the 20 students who answered, 15% considered the classes to be good, and 85% excellent, but there was also a regular, bad and terrible option, but no votes were registered in these. Regarding the performance of teachers, 95% considered them to be excellent and 5% good. When the question was about the teachers' level of knowledge, 100% thought that they knew very well what was being taught. However, only 80% gave the maximum score when asked if the teachers knew how to illustrate and explain the concepts well, with the remaining 20% choosing the second best evaluation.

A score of 1 to 5 was also asked to be given for the following statement: "I feel like I know more about computational thinking". 65% scored maximum, 25% scored 4, 1 student scored 3 and another student scored 2; nobody gave the worst evaluation.

Regarding the students' self-perception about their skills acquired during the course, the following statement was made: "If I have the idea for a game or program, I would be able to do it; It's only a matter of time". In this case, 60% gave the highest score, 20% the second highest, and the other 20% gave the third highest score.

65% of students thought that 90 minutes a week was sufficient, with the rest of them thinking they could have more.

What stood out most of the entire questionnaire, however, were the responses to the statement "I continued to make my games and projects at home". Only 30% did it frequently, most (40%) did it from time to time, 15% did it almost never and 15% did not. This reflected in the perception of those parents, who sought a utilitarian and immediate approach to the course, understanding that their children should come home and want to program frequently after the course. However, although it was not included in the questionnaire, informal conversations

TABLE I
PARENTS' FEEDBACK AT THE END OF THE COURSE IN 2017

Statement	Grade 1	Grade 2	Grade 3	Grade 4	Grade 5
I feel like my son enjoyed the classes.	0	0	12,5%	37,5%	50%
I feel like my son liked the teachers.	0	0	0	37,5%	62,5%
I feel like my son knows more about computational thinking.	0	0	25%	25%	25%
I found the course relevant to my son.	0	0	37,5%	12,5%	50%
My son showed me what he was doing in the classroom.	0	25%	0	62,5%	12,5%
I felt that my son was being taught by good professionals.	0	0	0	37,5%	62,5%

with parents and students identified that few had access to any type of computer at home, if they wanted to continue with their studies.

It is also important to add that the methodology, even in its first moments, never made use of "homework", since the given content should not be seen as an obligation but as a pleasant activity.

An evaluation form was also sent to 40 parents. However, we obtained only 8 feedbacks (20%), since it was not filled out *in loco*. Table I shows the statements made and the answers that were given.

An observation to be added is that, although it was explained to the parents at the beginning of the course, there is no way to guarantee that they really know the meaning of "computational thinking", which threatens the validity of the answers to this statement.

Responses related to children's proactivity in showing their work to parents corroborate the statistics of students who did not work at home. However, it is possible that the parents who answered the form were part of those who had a more positive perception of the course, perhaps because they saw what their children produced.

At the end of 2017, it was identified that the use of a purely script-based language, such as Python, would not be ideal for teaching computational thinking to pre-teens and teenagers, even if they had an introduction to programming using Scratch. The gap in level of abstraction between these tools was very wide. Because of this, a smoother transition was planned for the following year, starting with Scratch, followed by *GDevelop*[3], an open source alternative to *Construct*[4], one of the best-known game creation tool.

The result of applying such a tool, from an observational point of view, was that the students liked it, but part of it was just reproducing what the teacher presented to them. Which brings a second important point to the development of the methodology: the teachers.

There is no use in a proposed methodology, without the proper training of the professionals who will put it into practice. Didactic are also essential in this process, especially to maintain the level of engagement [48].

The most significant dropout of students was noted when the year turned. However, the longest stay was for students under 10 years old (58%). This produced a hypothesis that perhaps this age group was more willing to the subject of computational thinking. It was evident their blooming creativity, as well as a more genuine excitement. Pre-teens and teenagers were more apathetic in comparison.

A game jam, which is a meeting of developers to create a game, was proposed to all classes in 2018. The big difference is that they would only have 90 minutes to present a result. This ended up helping to identify that children under 10 years old were more focused when in pairs, or trios, differently when they were working individually on their computers.

These perceptions were fundamental to the new change in methodology, which, in the following year, began to focus specifically on children aged 5 to 10 years old. With 54 students, in 2019, classes started to have 60 minutes for children aged 8 to 10 years old, and 40 minutes for children aged 5 to 7. This is because the level of attention of a younger child was lower, in addition to the importance of not leaving children with a possible feeling of boredom at the end of class. This was thought based on the peak-end rule [78], which says that the emotional recollection of an experience is always based on the average between the peak and the end of it.

Another change applied to the methodology was a more specific segmentation, with children aged 8 to 10 years using *Scratch* on computers, and the newer ones with tablets and unplugged computing exercises.

The use of project-based learning was used about a year after the application of the computational thinking course began. The reason is that it serves as a teaching method in which students acquire knowledge and skills working for a certain period to solve a challenge that is complex for them [79]. However, it was clear that game creation activities lasting 3 weeks or more, for *Scratch* students and other development tools, were discouraging them. It was assumed, then, that for younger children, this would turn out to be even less effective.

Thus, the methodology that started to be developed was based on competencies that need to be worked on throughout the course, based on four of the main pillars of computational thinking: abstraction, pattern recognition, problem decomposition and algorithmic thinking.

Currently, three segments were created to develop the methodology:

- Digital Creativity, for children not yet literate - contains 22 activities,
- Computational Thinking, for literate children up to 7 years old - contains 23 activities,

[3]https://gdevelop-app.com
[4]https://www.construct.net/en

- Scratch, for children between 8 and 10 years old - contains 18 projects.

At the moment, only part of these activities are unplugged:

- Digital Creativity - **77% unplugged**,
- Computational Thinking - **78% unplugged**, and
- Scratch - **there is no unplugged activity**.

The intention, however, is that the course can be carried out only, or preferably, with unplugged activities, especially for the younger age groups, since the idea is to take this type of education to public schools that do not necessarily have a technological infrastructure that supports the teaching of computational concepts.

As an example of unplugged activity created for the methodology in question, we have "The Robot and the Programmer". In it, two people participate, one being the robot and the other the programmer. The robot, initially the teacher, must follow the commands of the student, who is the programmer. These commands are of the type:

- Turn right,
- Turn left,
- Walk forward,
- Backward, etc.

The robot's goal may be to draw something on the board, or to pick up an object.

The application of this activity allows the computer and other technological devices to be demystified. At the same time, the concept of algorithmic thinking is being consolidated in the minds of children. It is remarkable to see how much fun they have in the learning process with such an activity. This is mainly due to the inversion of expectation in simple situations such as when it is said "take the pen" by the programmer, and the teacher, in this case, the robot, closes his hand without moving.

Below are the conclusions about the work developed to date.

V. Conclusion

Analyzing a computational thinking course over 7 semesters, it was possible to identify an improvement in development skills with the use of computational tools such as Scratch and GDevelop. However, there is still no measurement to know quantitatively, the level of improvement in the computational thinking of children between 5 to 10 years old.

In addition, although there has been a self-perception by parents and students about an increase in their own knowledge of computational logic, it is necessary that a formal assessment be made before the course starts, 6 months after its beginning and another 6 months after that. Another alternative would be to assess students' performance in STEAM-related disciplines, such as mathematics, before and after applying the course. However, this leads to another problem, since for a more adequate perception of the benefits of teaching computational thinking, it is necessary to implement this content in the curricular matrix of the educational institution, integrated with the other subjects.

A second application would aim to search for alumni and take tests to identify how much of the knowledge of computational thinking remained after the course ended.

It should also be highlighted the problem of student engagement, presented in the theoretical foundation of this article. However, the public education scenario has a greater engagement by students [80], which makes the opportunity to develop this work with this target audience favorable.

The most current version of the methodology, used in 2019, which focuses on activities independent of digital technologies, needs to be able to develop the pillars of creative, scientific and computational reasoning. And this must be developed in the early years of the child's life, to ensure greater sedimentation of this type of knowledge.

At the same time, in addition to focusing on the student, creating an appropriate training for the teacher proved to be fundamental since the socio-emotional skills to deal with children and adolescents need to be developed, at the risk of damaging the project of teaching the discipline as a whole.

Also, there must be an attempt to create mechanisms that reduce the weight of the environment where such methodology will be applied. This means that this type of content should minimize the interference of physical spaces and externalities. This will facilitate the dissemination of this type in Brazilian public schools, democratizing the teaching of computational thinking.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

[1] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006, publisher: ACM New York, NY, USA.

[2] P. Seymour, "Mindstorms; Children, Computers and Powerful Ideas," *New York: Basic Book*, 1980.

[3] R. S. Monclar, M. A. Silva, and G. Xexéo, "Jogos com Propósito para o Ensino de Programação," *17th Brazilian Symposium on Games and Digital Entertainment (SBGames 2018)*, 2018.

[4] M. Prensky, "H. sapiens digital: From digital immigrants and digital natives to digital wisdom," *Innovate: journal of online education*, vol. 5, no. 3, 2009.

[5] A. Raabe, A. F. Zorzo, and P. Blikstein, *Computação na Educação Básica: Fundamentos e Experiências*. Penso Editora, 2020.

[6] E. Dearing, H. Kreider, S. Simpkins, and H. B. Weiss, "Family involvement in school and low-income children's literacy: Longitudinal associations between and within families." *Journal of Educational Psychology*, vol. 98, no. 4, p. 653, 2006.

[7] T. W. House, "President Obama Is the First President to Write a Line of Code," Dec. 2014, library Catalog: obamawhitehouse.archives.gov. [Online]. Available: https://obamawhitehouse.archives.gov/blog/2014/12/10/president-obama-first-president-write-line-code

[8] M. Resnick, "All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten," in *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, 2007, pp. 1–6.

[9] P. Cruz, O. N. Filho, G. Corrêa, and T. Pereira, "Educação Já: Uma proposta suprapartidária de estratégia para a Educação Básica brasileira e prioridades para o Governo Federal em 2019-2022," *Interesse Nacional*, 2018.

[10] U. Economia, "O que é o teto de gastos, como é definido e por que causa tanta polêmica - 05/09/2019 - UOL Economia." [Online]. Available: https://economia.uol.com.br/noticias/redacao/2019/09/05/teto-de-gastos-constituicao-calculo-limite-gastos-publicos.htm%3E

[11] R. Whyte, S. Ainsworth, and J. Medwell, "Designing for Integrated K-5 Computing and Literacy through Story-making Activities," in *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 2019, pp. 167–175.

[12] J. M. Wing, "Five deep questions in computing," *Communications of the ACM*, vol. 51, no. 1, pp. 58–60, 2008, publisher: ACM New York, NY, USA.

[13] ——, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717–3725, 2008, publisher: The Royal Society London.

[14] ——, "Computational thinking benefits society," *40th Anniversary Blog of Social Issues in Computing*, vol. 2014, 2014.

[15] J. Lockwood and A. Mooney, "Computational Thinking in Education: Where does it fit? A systematic literary review," *arXiv preprint arXiv:1703.07659*, 2017.

[16] J. Voogt, P. Fisser, J. Good, P. Mishra, and A. Yadav, "Computational thinking in compulsory education: Towards an agenda for research and practice," *Education and Information Technologies*, vol. 20, no. 4, pp. 715–728, 2015, publisher: Springer.

[17] P. Phillips, "Computational thinking: A problem-solving tool for every classroom," *Communications of the CSTA*, vol. 3, no. 6, pp. 12–16, 2009.

[18] P. B. Henderson, T. J. Cortina, and J. M. Wing, "Computational thinking," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, 2007, pp. 195–196.

[19] M. Guzdial, "Education Paving the way for computational thinking," *Communications of the ACM*, vol. 51, no. 8, pp. 25–27, 2008, publisher: ACM New York, NY, USA.

[20] I. Lee, "Computer science: Critical K-8 learning," *Special Issue Computer Science K-8: Building a Strong Foundation*, pp. 10–11, 2012.

[21] J. L. Ramos and R. G. Espadeiro, "Os futuros professores e os professores do futuro. Os desafios da introdução ao pensamento computacional na escola, no currículo e na aprendizagem," *Educação, Formação & Tecnologias-ISSN 1646-933X*, vol. 7, no. 2, pp. 4–25, 2014.

[22] S. Grover and R. Pea, "Computational thinking in K–12: A review of the state of the field," *Educational researcher*, vol. 42, no. 1, pp. 38–43, 2013, publisher: Sage Publications Sage CA: Los Angeles, CA.

[23] T. Bell, F. Rosamond, and N. Casey, "Computer Science Unplugged and Related Projects in Math and Computer Science Popularization," in *The Multivariate Algorithmic Revolution and Beyond*, H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7370, pp. 398–456, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-30891-8_18

[24] P. Curzon, P. W. McOwan, Q. I. Cutts, and T. Bell, "Enthusing & inspiring with reusable kinaesthetic activities," in *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, 2009, pp. 94–98.

[25] B. Clarke, *Computer Science Teacher: Insight Into the Computing Classroom*. Swindon, UK: BCS, The Chartered Institute for IT, Aug. 2017.

[26] O. Hazzan, T. Lapidot, and N. Ragonis, *Guide to Teaching Computer Science*. London: Springer London, 2011. [Online]. Available: http://link.springer.com/10.1007/978-0-85729-443-2

[27] A. Tucker and others, *A model curriculum for K–12 computer science: final report of the ACM K-12 task force curriculum committee (2003)*. Association for Computing Machinery, New York, NY, United States, 2017.

[28] E. Schofield, M. Erlinger, and Z. Dodds, "MyCS: A CS Curriculum for Middle-Years Students," *J. Comput. Sci. Coll.*, vol. 29, no. 4, pp. 145–155, Apr. 2014, place: Evansville, IN, USA Publisher: Consortium for Computing Sciences in Colleges.

[29] J. Goode and J. Margolis, "Exploring computer science: A case study of school reform," *ACM Transactions on Computing Education (TOCE)*, vol. 11, no. 2, pp. 1–16, 2011, publisher: ACM New York, NY, USA.

[30] K. Falkner and R. Vivian, "A review of computer science resources for learning and teaching with K-12 computing curricula: An Australian case study," *Computer Science Education*, vol. 25, no. 4, pp. 390–429, 2015, publisher: Taylor & Francis.

[31] T. C. Bell, I. H. Witten, and M. Fellows, *Computer Science Unplugged: Off-line activities and games for all ages*. Citeseer, 1998.

[32] T. Bell and H. Newton, "Unplugging computer science," in *Improving computer science education*. Routledge, 2013, pp. 75–90.

[33] C. Unplugged, "CS Unplugged," 2020. [Online]. Available: https://csunplugged.org/en/

[34] T. Bell, P. Curzon, Q. Cutts, V. Dagiene, and B. Haberman, "Overcoming obstacles to CS education by using non-programming outreach programmes," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2011, pp. 71–81.

[35] J. Hromkovič and R. Lacher, "The computer science way of thinking in human history and consequences for the design of computer science curricula," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2017, pp. 3–11.

[36] E. Prieto-Rodriguez and R. Berretta, "Digital technology teachers' perceptions of computer science: It is not all about programming," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, 2014, pp. 1–5.

[37] T. Bell and J. Vahrenhold, "CS Unplugged—How Is It Used, and Does It Work?" in *Adventures Between Lower Bounds and Higher Altitudes*. Springer, 2018, pp. 497–521.

[38] N. Koblitz, "Crypto galore!" in *The Multivariate Algorithmic Revolution and Beyond*. Springer, 2012, pp. 39–50.

[39] T. Bell, "A low-cost high-impact computer science show for family audiences," in *Proceedings 23rd Australasian Computer Science Conference. ACSC 2000 (Cat. No. PR00518)*. IEEE, 2000, pp. 10–16.

[40] H. Caldwell and N. Smith, *Teaching Computing Unplugged in Primary Schools: exploring primary computing through practical activities away from the computer*. Learning Matters, 2016.

[41] B. Vöcking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, and D. Wagner, *Algorithms unplugged*. Springer Science & Business Media, 2010.

[42] T. Nishida, S. Kanemune, Y. Idosaka, M. Namiki, T. Bell, and Y. Kuno, "A CS unplugged design pattern," *ACM SIGCSE Bulletin*, vol. 41, no. 1, pp. 231–235, 2009, publisher: ACM New York, NY, USA.

[43] B. Rodriguez, S. Kennicutt, C. Rader, and T. Camp, "Assessing computational thinking in cs unplugged activities," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 501–506.

[44] R. Thies and J. Vahrenhold, "On plugging" unplugged" into CS classes," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 365–370.

[45] J. D. Finn and D. A. Rock, "Academic success among students at risk for school failure." *Journal of applied psychology*, vol. 82, no. 2, p. 221, 1997, publisher: American Psychological Association.

[46] C. Brewster and J. Fager, *Increasing student engagement and motivation: From time-on-task to homework*. Northwest Regional Educational Laboratory Portland, OR, 2000.

[47] H. M. Marks, "Student engagement in instructional activity: Patterns in the elementary, middle, and high school years," *American educational research journal*, vol. 37, no. 1, pp. 153–184, 2000, publisher: Sage Publications.

[48] J. R. Jablon and M. Wilkinson, "Using engagement strategies to facilitate children's learning and success," *YC Young Children*, vol. 61, no. 2, p. 12, 2006, publisher: National Association for the Education of Young Children.

[49] P. Schletchy, *Shaking up the schoolhouse: How to support and sustain education innovation*. San Francisco, CA: Jossey-Bass, 2001.

[50] S. Graham and B. Weiner, "Theories and principles of motivation," *Handbook of educational psychology*, vol. 4, pp. 63–84, 1996.

[51] R. D. Felner, A. W. Jackson, D. Kasak, P. Mulhall, and others, "The impact of school reform for the middle years," *Phi Delta Kappan*, vol. 78, no. 7, p. 528, 1997, publisher: Phi Delta Kappa.

[52] R. Rossi and A. Montgomery, "Educational reforms and students at risk: A review of the current state of the art," *Washington, DC: US Department of Education, Office of Educational Research and Improvement, Office of Research. Retrieved May*, vol. 25, p. 2007, 1994.

[53] P. A. McDermott, M. Mordell, and J. C. Stoltzfus, "The organization of student performance in American schools: Discipline, motivation, verbal learning, nonverbal learning." *Journal of Educational Psychology*, vol. 93, no. 1, p. 65, 2001, publisher: American Psychological Association.

[54] G. Yair, "Reforming motivation: How the structure of instruction affects students' learning experiences," *British educational research journal*, vol. 26, no. 2, pp. 191–210, 2000, publisher: Taylor & Francis.

[55] J. I. Goodlad, *A place called school. Prospects for the future.* ERIC, 1984.

[56] M. Roderick and M. Engel, "The grasshopper and the ant: Motivational responses of low-achieving students to high-stakes testing," *Educational Evaluation and Policy Analysis*, vol. 23, no. 3, pp. 197–227, 2001, publisher: Sage Publications Sage CA: Los Angeles, CA.
[57] T. W. Long and M. K. Gove, "How engagement strategies and literature circles promote critical response in a fourth-grade, urban classroom," *The Reading Teacher*, vol. 57, no. 4, pp. 350–361, 2003.
[58] L. M. Tabuti and R. Nakamura, "Métodos para o Desenvolvimento de Jogos Digitais de Lógica: Uma Revisão Sistemática," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 26, 2015, p. 41, issue: 1.
[59] A. Harrison, T. Hulse, D. Manzo, M. Micciolo, E. Ottmar, and I. Arroyo, "Computational thinking through game creation in STEM classrooms," in *International Conference on Artificial Intelligence in Education*. Springer, 2018, pp. 134–138.
[60] M. L. Wu and K. Richards, "Facilitating computational thinking through game design," in *International Conference on Technologies for E-Learning and Digital Entertainment*. Springer, 2011, pp. 220–227.
[61] A. Repenning, D. Webb, and A. Ioannidou, "Scalable game design and the development of a checklist for getting computational thinking into public schools," in *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 265–269.
[62] D. Pérez-Marín, R. Hijón-Neira, A. Bacelo, and C. Pizarro, "Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?" *Computers in Human Behavior*, p. 105849, 2018, publisher: Elsevier.
[63] M. Weigend, J. Vaníček, Z. Pluhár, and I. Pesek, "Computational Thinking Education through Creative Unplugged Activities," *Olympiads in Informatics*, vol. 13, pp. 171–192, Jul. 2019. [Online]. Available: https://ioinformatics.org/page/ioi-journal-index/44#volume13
[64] J. Arthur, *Research methods and methodologies in education*. Sage publications, 2012.
[65] B. David, K. Werbin, and S. Shaw, "Integrated development and production tools for building hypermedia courseware and interactive scenarios," in *Proceedings of ED-MEDIA/ED-TELECOM*, vol. 97, 1997, pp. 241–246.
[66] M. Pivec, O. Dziabenko, and I. Schinnerl, "Aspects of game-based learning," in *3rd International Conference on Knowledge Management, Graz, Austria*, 2003, pp. 216–225.
[67] R. C. HAYDT, "Curso de didática geral. 8ª," *Edição. São Paulo: Ática*, 2006.
[68] K. S. Tekinbas, R. Torres, L. Wolozin, R. Rufo-Tepper, and A. Shapiro, *Quest to learn: Developing the school for digital kids*. MIT Press, 2010.
[69] B. S. Bloom and others, "Taxonomy of educational objectives. Vol. 1: Cognitive domain," *New York: McKay*, pp. 20–24, 1956.
[70] J. P. Greene, *High school graduation rates in the United States*. Black Alliance for Educational Options Washington, DC, 2001.
[71] A. Lenhart and M. Madden, *Teen content creators and consumers*. Pew Internet & American Life Project Washington, DC, 2005, vol. 2.
[72] H. Jenkins, *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press, 2009.
[73] IDEB, "IDEB – Planilhas com resultados por escola - Artigo - INEP," 2018. [Online]. Available: http://portal.inep.gov.br/artigo/-/asset_publisher/B4AQV9zFY7Bv/content/id/1511536
[74] P. K. Honorato and A. Manarin, "Tecnologia e Educação: o que pensam os professores?" 2017, library Catalog: www.todospelaeducacao.org.br. [Online]. Available: https://www.todospelaeducacao.org.br/tecnologia/
[75] J. McGonigal, *Reality is broken: Why games make us better and how they can change the world*. Penguin, 2011.
[76] G. Zichermann and C. Cunningham, *Gamification by design: Implementing game mechanics in web and mobile apps*. " O'Reilly Media, Inc.", 2011.
[77] K. Werbach and D. Hunter, *For the win: How game thinking can revolutionize your business*. Wharton Digital Press, 2012.
[78] D. Kahneman, *Rápido e devagar: duas formas de pensar*. Objetiva, 2012.
[79] J. W. Thomas, "A review of research on project-based learning," *The Autodesk Foundation*, 2000.
[80] P. N. d. Fonsêca, B. d. J. Lopes, R. M. Palitot, A. M. Estanislau, R. N. Couto, and G. L. d. H. Coelho, "Engajamento escolar: explicação a partir dos valores humanos," *Psicologia escolar e educacional*, vol. 20, no. 3, pp. 611–620, 2016, publisher: SciELO Brasil.