Article

Educational Challenges for Computational Thinking in K–12 Education: A Systematic Literature Review of "Scratch" as an Innovative Programming Tool

Hugo Montiel * and Marcela Georgina Gomez-Zermeño

School of Humanities, Tecnologico Monterrey, Monterrey 64849, Mexico; marcela.gomez@tec.mx
* Correspondence: a00600529@itesm.mx; Tel.: +52-812-354-4705

Abstract: The use of information and communications technologies (ICTs) has emerged as an educational response amidst the COVID-19 pandemic, providing students the technological tools that enable them to acquire or strengthen the necessary digital skills to develop computational knowledge. The purpose of this study was to analyze Scratch, a programming language used to foster the teaching of computational thinking, particularly in K–12 education. A systematic literature review (SLR) was conducted, identifying 30 articles on the topic of Scratch and computational thinking in the database ProQuest Central from January 2010 to May 2020. These articles were analyzed to identify the use of Scratch worldwide and the educational impact it has on computational thinking, specifically in K–12 education. The results highlight the following: (1) countries which incorporated Scratch into their teachers' study plans (curricula); (2) the transformation of learning environments that Scratch promotes; and (3) the importance of incorporating tools like Scratch in the current curricula and, more importantly, developing the framework for innovative ICTs capable of transforming education.

Keywords: computational thinking; educational innovation; K–12 education; project-based research; Scratch; teacher training; higher education

Citation: Montiel, H.; Gomez-Zermeño, M.G. Educational Challenges for Computational Thinking in K–12 Education: A Systematic Literature Review of "Scratch" as an Innovative Programming Tool. *Computers* **2021**, *10*, 69. https://doi.org/10.3390/computers10060069

Academic Editors: Antonio Sarasa Cabezuelo, Covadonga Rodrigo San Juan and Santi Caballé

Received: 14 April 2021
Accepted: 17 May 2021
Published: 21 May 2021

1. Introduction

It is a reality that the COVID-19 pandemic brought socioeconomic disruptions and technological changes worldwide. Political, social, religious, and sporting events were canceled to promote social distancing and prevent the virus from spreading widely. The educational sector is no exception, and institutions had to take measures based on their human capabilities and technological resources. Some institutions decided to finish their ongoing school terms abruptly, while others adapted their operations to the sanitary requirements forced by the pandemic. In general, the coronavirus pandemic presents a significant challenge to teachers at all educational levels. This hurdle demands the development of new competencies in using computational tools and the continuous adaptation of digital pedagogical strategies to meet the students' needs.

As this digital transformation has advanced, the skills that students must possess have been changing along with emerging technologies. Countries feel the need to make changes to their educational systems to ensure that students acquire these new skills [1]. In K–12 education, computational thinking (CT) is an important part of cultivating students' key abilities [2]. Lonka mentions "that students should learn to identify the central principles and practices of programming and understand how it affects everyday life" [3]. Despite its recent acclaim, there have been some drawbacks and uncertainty surrounding CT regarding teacher training and development for understanding the aims and intentions of CT education [4].

Contributions such as Zhang and Nouri's SLR on Scratch have laid the foundation for the relationship between Scratch, problem solving, computing, and programming at the K–9 level [5]. Other studies, like the systematic literature review on computational thinking with Scratch by Fagerlund et al. [4], emphasize the concepts associated with CT and CT fostering in primary education while acknowledging the importance of refining and validating meaningful ways to assess CT in students' projects and programming processes. Moreno-León and Robles's review [6] presents studies on the use of Scratch for enhancing and developing not only CT skills, but competencies and capabilities beyond programming or coding skills. Considering the aforementioned reviews as a springboard, this study intends to review and summarize the literature about the use of Scratch as a digital tool to promote CT in K–12 education. The analysis can serve as one of the catalysts for the digital transformation that has been forced by COVID-19.

The results of this review intend to show how countries around the world have incorporated CT not only into their students' curricula but, more importantly, into formative plans for teachers. Understanding what features make Scratch a feasible tool to teach CT while innovating and renewing the traditional learning environment is of the utmost importance, given the disruption in the educational setting. Motivated by the shift in pedagogical strategies and the introduction of computational thinking as an essential skill to be developed in primary education, we aim to evaluate the impact Scratch has on learning environments and how it could possibly encourage the development of teacher training frameworks to support CT learning. The study also strives for answers to the questions concerning whether Scratch could serve as a digital answer to the challenges and difficulties presented by COVID-19, supporting the idea of the development of technological competencies among educators to be better prepared for future contingencies or disruptions to education.

1.1. Computational Thinking as an Educational Challenge

Information and communications technologies (ICTs) are more present than ever in the educational setting, although this was not always the case. At the end of the 1970s, an official report emphasized the importance of differentiating between informatics as a subject and informatics as a teaching and learning tool [7]. This report prompted, at the time, the creation of an elective informatics course, whose goal was not to teach programming languages but to promote an algorithmic way of thinking that would be useful for other subjects [8]. This way of thinking led to the belief that informatics should not be taught as a discrete subject; rather, it should be learned and mastered in different disciplines supported by the integration of ICT tools. Programming, which had long been a major concern in the world of computational education, was no longer considered accessible for everyone [9].

The 21st century brought initiatives that acknowledged the need for informatics skills to be taught and assessed in the learning environment. Associations like Public Education and Informatics, which brings together educators, researchers, and activists, have been stressing the urgent need to include informatics as a subject in schools from an early age [8]. This approach allowed new ICT tools and skills to emerge, with multiple benefits to the achievement of learning objectives [10].

One of these skills is CT which, according to Wing, "involves solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science" [11]. Fundamentally, CT involves breaking down complex problems into familiar and manageable subproblems, using a sequence of steps or instructions to provide a solution to set problems, reviewing the solution's transferability to similar problems, and finally, determining if a computer process is able to help solve those problems more efficiently [12]. In summary, the idea of CT is to allow students to develop an understanding of computing and programming competencies, thus allowing them to move from being users of technology to producers of information technology [13]. As this view had been accepted by many, in 2016, the International Society for Technology in Education (ISTE) [14], which developed the standards for teachers and students to use technology in teaching–learning processes, included CT in the basic skills to be acquired by students. Aiming for emancipation through training is undoubtedly a laudable goal, but it is very difficult to achieve. Technologies, in the guise of openness, too often reinforce social silos [10].

1.2. Scratch Used as an Innovative Educational Response

Scratch is a programming language that provides students with the means to create stories, animations, games, and music and to share projects on the web [15]. It is one of the emerging technologies that can be applied to interactive game design, storytelling, animation, and multimedia projects in the classroom [16].

Scratch scripts are created by putting together blocks that represent programming statements, expressions, and control structures. The shapes of the blocks suggest how they fit together, and the drag-and-drop system does not allow connecting blocks in ways that would have no effect on the programming logic [17]. This approach, known as block-based programming, is easy to understand because the blocks are described in common language which, combined with the aforementioned drag-and-drop interactions and the convenience of browsing programming languages, makes its mastery attainable [2]. Scratch's trial and error design allows students to play with situations, modify parameters on the fly, and tinker, all while coding without having to write an algorithm [18]. Scratch builds on the constructionist ideas of Logo and Etoys and provides a bridge to Brennan and Resnick's framework, which is likely to be suitable for CT in the programming contexts of K–12 education [19–25]. Scratch's popularity and userbase make it one of the predominant ITC tools to start teaching computational thinking (CT) to students from a young age.

It is not farfetched to assume that, given Scratch's features and capabilities, institutions would include it in their formative plans for teachers. Surprisingly, that is not the case, as mentioned by Fagerlund [4], who stated that there have been some shortcomings and uncertainty surrounding CT in terms of teacher training needs and the aims and intentions of CT education. There is a present need for frameworks and guides like Csizmadia's [26], which supports teaching CT concepts, approaches, and techniques in the classroom.

1.3. Project-Based Research to Develop Computational Thinking in Grades K–12

Moving toward the education of the future involves an extensive modernization of all educational processes. Such modernization includes the introduction of smart technologies, systems, and devices that create new opportunities for training organizations to have higher standards and employ innovative solutions. Education should be transformed rather than reformed to confront the digital challenges successfully. The key to this transformation is not to standardize education but to personalize it, discover the individual talents of each child, and place the students in an environment where they want to learn and naturally discover their true passions [27–29]. To perform such a transformation, educators have launched many initiatives to promote computer sciences and programming among the population, especially among children. Learning how to program a computer has many benefits for those who practice it, but the best one is that it helps people think about solving problems [28].

In France, the National Research Agency (ANR) promotes project-based research to stimulate innovation by promoting collaborative, multidisciplinary projects between the public and private sectors. Through the IE-CARE project, it was proposed to "set up sustainable modalities for informatics curricula in compulsory education" [30]. One of the main interests is to know how to articulate different content throughout the first three French school cycles progressively. This research project has "a multidisciplinary character, bringing together researchers in the humanities, social sciences, and informatics" [30]. It employs three main axes that depict a transversal task aimed at ensuring convergence in the research [30]. The first axis identifies teachable concepts and pedagogical practices. The second axis aims at designing, testing, and validating pedagogical scenarios and resources both in primary school and junior high school. The third axis "focuses on building a framework that supports teachers and trainers in informatics" [30].

The IE-CARE project "aims to develop a culture in informatics as well as a technical culture that includes professional development for primary school teachers and people in charge of supervising the teachers' actions" [30]. During the transversal task of the first and second axes for the extraction of teachable content and proposals for curricula and instruments, a systematic literature review (SLR) was conducted to identify studies related to Scratch being used as a tool to facilitate CT teaching in K–12 or K–9 educational settings. The following research question was proposed: Which elements should be considered in a teacher training framework to support learning computational thinking through Scratch in K–9 or K–12?

Before following the methodology of Brereton et al. [31] pertaining to SLRs, we divided our main objective into three sub-objectives which would guide our study. These sub-objectives are later addressed in the methods and materials section. It is pertinent to mention this, as these subjects of interest will serve as inclusion and exclusion criteria while reviewing the existing literature. The following section introduces the findings for each of the sub-objectives within the selected literature.

2. Results

As part of the Brereton et al. [31] methodology, the document review phase pertains to the creation and validation of a written report with the findings of the study, which are presented as the results related to the proposed research questions presented in the previous phase. The tools used for the illustrated figures were Vosviewer and Tableau.

2.1. RQ1: Scratch around the World in Formative Plans for Teachers in K–12 and K–9 Education as a Means to Develop Computational Thinking (CT)

Given the acceptance of CT as a necessary skill in the educational setting, we sought to find within the studies the countries that incorporated Scratch in the development and formative plans for their teachers [2]. Having these lessons incorporated into the curricula would raise the question of whether the teachers or students have the skills necessary to apply them.

The first study found was in Italy, where an empirical experiment was carried out with 141 Italian preservice teachers who attended a programming course. Demir reports that the program had the following aims: (1) use Scratch 2.0 to provide them the main coding concepts, (2) offer practical advice on how to design educational applications to be used in the school context, and (3) assess their applications by applying an already existing methodology for giving them feedback on their programming expertise and CT skills [32].

The impact Scratch has had in Turkey is evident because there are 484,137 users of Scratch in this country, which comprises 1.44 percent of Scratch users around the world (as of 20 February 2019). Turkey is second only to the USA in Scratch usage, according to Scratch's worldwide map [33]. The developments in coding education have positively contributed to the coding education in Turkey, and consequently, a course titled Information Technologies and Software has been incorporated in the curricula of grades five and above since the 2012–2013 academic year [33]. One of the studies selected was Uzunboylu et al. [34], which aimed to analyze the countries where coding training was integrated into their curricula and compare them to studies on coding training, particularly in Turkey.

Initiatives in the United Kingdom, New Zealand, and Denmark suggest an imperative movement toward training young minds in computing. The fact that countries are implementing more and more programs does not help the problem of the shortage of teachers to teach introductory computing [35]. Such is the case of the VELA project, implemented in a diverse urban district in the United States of America. The purpose of this project was to introduce four concepts to students, namely variables (V), expressions (E), looping (L), and abstraction (A) in a new pedagogical approach toward computational understanding. This project required teachers to participate in two days of professional development. Teachers learned about the VELA rationale and the principles guiding the curriculum design. They participated in roleplaying during the curriculum's enactment, engaged in hands-on activities to experience using the curriculum and reviewing lesson plans [36].

Israel, to assess whether creating games proved to be compelling for learning computational concepts and practices, conducted a study among seventh- to ninth-grade students in middle school for three years (2011–2013). The students were in a new CS curriculum that used Scratch. The results of that study led to changes in the teaching methods for loop concepts [37].

Greece decided to implement a program to help primary and secondary education teachers acquire a solid comprehension of the basic programming concepts which are common in all programming languages [29,38]. They decided to use Scratch because it is effective for introducing programming concepts. Trainee evaluations for this project were positive; the trainees indicated that they would like to participate in another learning activity with more advanced topics [39].

Many internationally funded programs can be found among the studies that include "Code Week", "Hour of Code", and "Computer Science for All" [40]. These programs intend to reinforce the coding skills of students and teachers using a mixture of pedagogical approaches and technologies [39,41,42]. Figure 1 and Table 1 illustrate the countries and initiatives that incorporate Scratch in their formative plans, as well as the occurrence of mentions for each country throughout the studies.

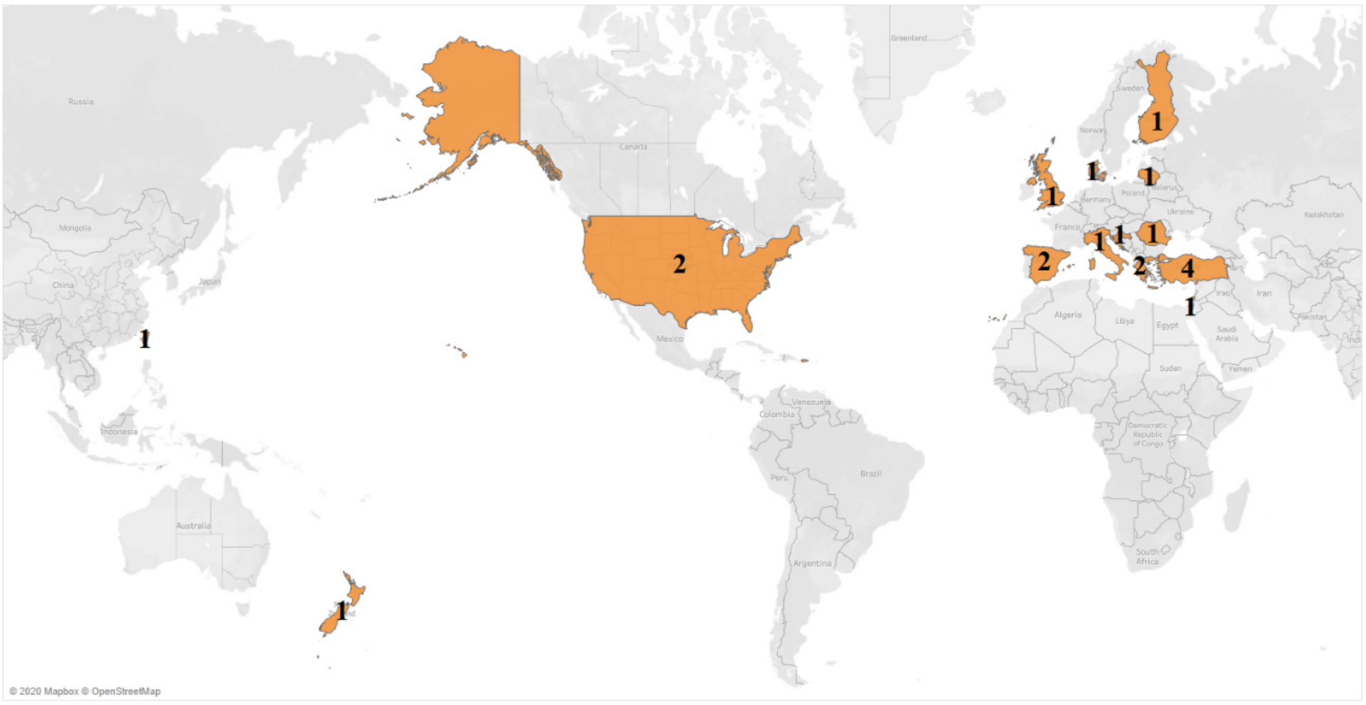

Figure 1. Countries that incorporate Scratch in formative plans.

2.2. *RQ2: Features of Scratch to Assist in Teaching CT in K–12 and K–9 Education*

The second aspect we considered in the study pertained to identifying the features mentioned in the literature that described Scratch as a feasible tool for CT instruction. We decided to organize the different elements of Scratch mentioned in the studies into categories.

The most prominent category among the studies concerned the problem solving, logical thinking approach that Scratch uses. The Scratch members learn their problem solving and project design skills (e.g., logical reasoning and debugging problems), along with specific programming concepts (e.g., sequence, looping, conditional statements, variables, arrays, and Boolean logic) as they create digital media in Scratch [6,16,43–45]. This approach allows students to acquire programming skills, as Ali Oluk et al. [1] stated that individuals need to find different solutions to problems and select the fastest path. Finding the fastest path requires understanding the logic of the algorithm well. Çakıroğlu et al. [46] concurred with Hagge [43] and stated that "researchers believe students should acquire procedural, conditional, and analogical thinking skills in the programming process, which will allow students to assess the thought process of the problem as well".

Table 1. Countries' initiatives regarding CT and Scratch.

Country	Description
Turkey	Information technologies and Software incorporated into fifth grade curricula. Second biggest Scratch user in the world.
Italy	Scratch was incorporated into programming courses to assess programming expertise and CT skills of preservice teachers.
United Kingdom	Formulated computing curriculum as a result of the Royal Society policy charter.
New Zealand	Implemented the Computer Science Unplugged project, which included Scratch activities.
United States	VELA project targeted teachers to introduce programming concepts and computational understanding.
Israel	Introduced CT into school curriculum to further develop Israeli high-tech industry.
Greece	Program based on Scratch to introduce basic programming concepts to educators.

The second commonality we discovered among the studies concerned the social dynamic that Scratch provides. With over 21 million registered users, Scratch is the largest online programming community for youth [43]. Therefore, Scratch can provide students a social platform to share their creations. The "do it yourself" (DIY) culture embodied in Scratch is reflective of a trend in the literacy practices of youth to shift toward DIY media [47]. This increase in DIY media aligns with the "maker movement" defined by Halverson and Sheridan [48]. A growing number of people who engage in the creative production of artifacts find physical and digital forums to share their processes and products with others. The idea of "computational participation" is compelling as a social motivator to learn computing, especially for younger learners [35,38,41,42,49].

Block-based programming was another common denominator in the studies analyzed. As fifth- and sixth-grade students are mostly still in the pre-formal phase of cognitive development, programming games in visual programming languages offer them real experience in learning complex programming concepts [6,35,44]. Visual programming blocks allow syntax problems to be eliminated, thus permitting students to focus on algorithms [2,50]. Lee [51] even compared visual programming blocks to "putting together jigsaw puzzles or LEGO pieces, using a computer mouse instead of typing program-language constructs on a computer keyboard" [16,37,50,51]. These ideas compliment Traylor's statement of new technologies having a "low floor" and a "high ceiling", making it easy for people to get started while also allowing advanced users to do more complex things as well [25,41]. Although some authors like Hagge and Deng think that this approach limits the coding schema required for children to create projects, they acknowledge that one key goal should be to increase accessibility to coding concepts [2,43].

Some other concepts were mentioned throughout the studies, but the three described above were the most numerous.

2.3. RQ3: Scratch and Its Impact on the Design of Learning Environments and Teaching Resources in K–12 and K–9 Education

Our last research question aims to collect information regarding how Scratch helps in the design of learning environments in K–12 and K–9 education. Scratch was used in different ways throughout the studies as a tool to strengthen lesson plans and educational resources. Lee states that Scratch can provide an excellent platform for education researchers and practitioners to develop creative, enjoyable, interdisciplinary curriculum materials, becoming a creative medium for students to express their imagination and making school subjects considered boring and difficult more meaningful and engaging [41,42,51]. Scratch helps elementary learners create projects such as animations, games, and simulations amusingly, turning programming into a pleasurable, visual learning experience rather than a textual struggle [50,52].

Scratch is also found to foster subskills of computational thinking skills, namely creative thinking, algorithmic thinking, critical thinking, collaborative learning, and communication skills, which are similar to the skills expected from today's students [6,38,49,53]. The task-based learning environments provided by Scratch enhance motivation and cognitive outcomes [54]. As an example, the progression of early computational thinking model by Seiter and Foreman [55] can be considered useful for establishing age-appropriate curricula and defining lesson plans aligned with the students' cognitive development stages [32,55].

Some studies showed that teachers had positive experiences while incorporating Scratch and were eager to find ways to integrate CT into their lesson plans seamlessly [40]. There are several benefits to incorporating programming and simulations in the classroom. Simulations are effective at promoting scientific understanding. Many students are visual learners, and allowing them to visualize what they are being taught can help them remember it [43,56,57].

Specifics about the social features of Scratch mentioned in the studies include different tasks that involve creating a project, testing it, demonstrating it to the entire class, documenting and writing reflections on it, and playing with each other's games in the online studio [35]. Scratch emerged with an implicit belief that if learners could share and show some of their accomplishments, they would learn well and learn more. No longer merely recipients of knowledge, children create and publish an array of texts as they interact in virtual social spaces. As youths explore new ways to communicate and engage in an increasingly globalized world, the ways they find to make meaning expand [40,43].

Nowadays, educators lean toward designing active and dynamic learning environments based on instructional methodologies centered around the student. This new environment should be aligned with the availability and use of technologies in the classroom [58]. One common thought shared in the studies was the idea that the implementation of tools like Scratch motivates institutions to rethink and redesign instructional approaches and plans.

To further strengthen the findings on the studies, a density visualization of the most-mentioned concepts amongst the articles is illustrated in Figure 2. The concepts were limited to no less than 100 occurrences in all 30 articles. Entries such as student (1358), problem (646), programming (665), skill (500), process (348), computational thinking (328), program (280), block (259) were among the most mentioned concepts. The most-mentioned concepts are presented in Table 2.

Table 2. Concepts mentioned throughout the studies.

No.	Concept	Mentions
1	Student	1358
2	Problem	646
3	Programming	665
4	Skill	500
5	Process	348
6	Computational Thinking	328
7	Program	280
8	Block	259
9	Variable	178
10	Instruction	124

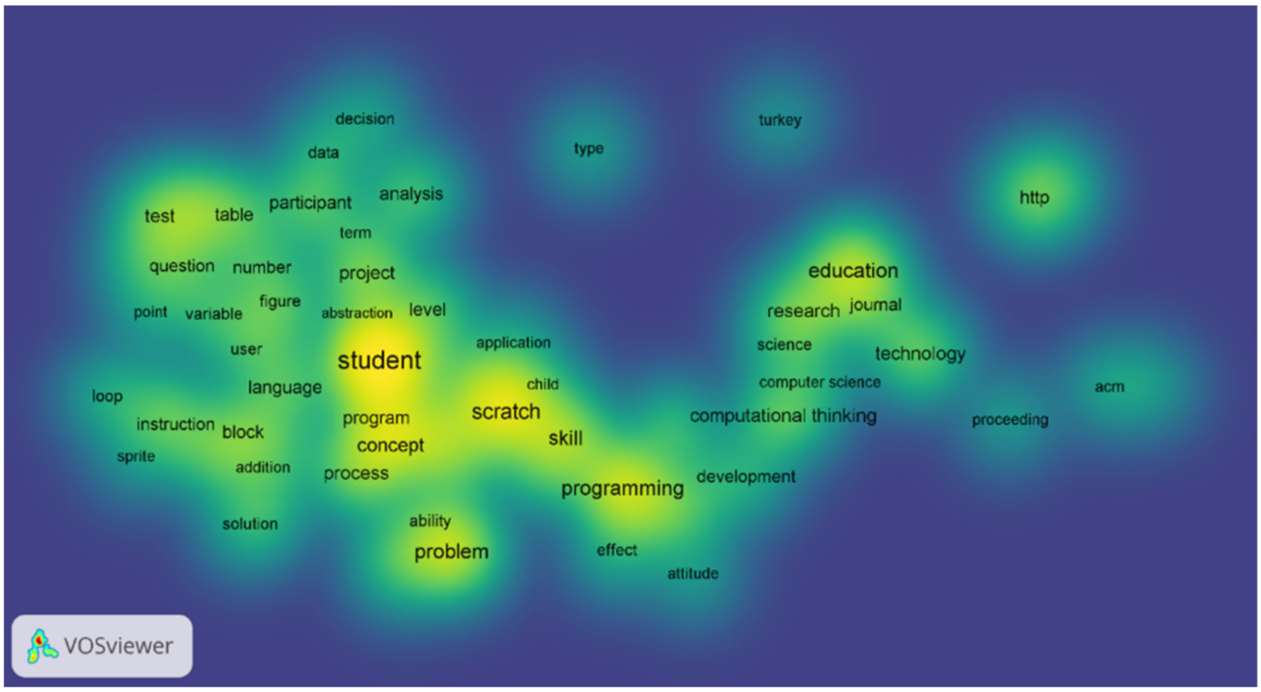

Figure 2. Density visualization of the frequency of concepts on Scratch (threshold of 100 occurrences, displaying 54 concepts).

3. Discussion

As the SLR was conducted, information was generated to answer the research question: Which elements should be considered in a teacher training framework to support learning computational thinking through Scratch in K–9 and K–12 education?

The rising popularity today of CT as a requisite skill for students has pushed countries to design, implement, and incorporate CT in the formative plans of the teachers' curricula as an element to be considered. The SLR results showed that the evaluation phase some countries conducted to determine Scratch's feasibility to teach CT concluded with leading the way into the implementation phase, where long-term formative assessments were designed to cope with the students' need for CT. Therefore, there is a need to provide teachers an understanding of the role CT plays not only in problem solving but also in other cognitive domains. The implementation of teacher training programs should create an educational path to teach, include, and assess CT abilities [32].

Scratch's features, reviewed throughout the SLR, suggest that the symbiotic relationship with the ever-changing educational paradigms and settings is another element to be considered. The shift in learning environments toward a DIY approach correlates with the self-paced dynamic Scratch presents. This, in turn, helps transition the role of the teacher into a more passive facilitator who guides the student into acquiring knowledge through the use of technology. Scratch nurtures the creation of such environments by incorporating technology in a gamified environment that results in students learning and fostering skills other than just CT. Creating inspiring and motivational learning environments for students has become of paramount importance and needs to be considered in a teacher training framework.

Some studies, like Hagge's [43], concur that "the vacuum-controlled environment Scratch provides limits the ways students can solve a problem and restricts their creativity; nonetheless, most studies agree that at these stages of primary education, it is more important to introduce coding concepts at the expense of flexibility". It is necessary to consider that, restricted or not, Scratch encourages the development of creativity and problem solving via the implementation of different routes or solutions to the same problem. It is also necessary to carefully consider how teachers use the affordances of computing tools and not merely put them in front of their students [44]. These observations bolster the argument that classroom management and course design by the teachers are crucial. Teachers should apply CT environments in their multidisciplinary teaching practice and guide students to transfer this thinking into multiple fields [2].

4. Materials and Method

An SLR was conducted to provide insights for a teacher training framework to support learning computational thinking through Scratch in K–12 and K–9 educational settings. Following the principles stated by Brereton et al. [31], this review will serve as a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. We grouped the SLR activities into three main phases (see Figure 3): (1) plan review, (2) conduct review, and (3) document review. [31,59].

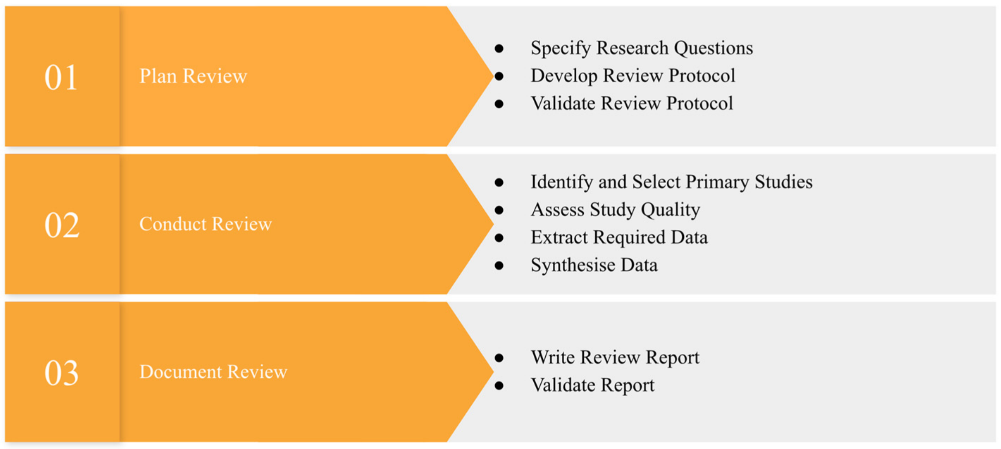

Figure 3. SLR phases, as suggested by Brereton et al. [31].

4.1. Plan Review

The first phase of the SLR methodology described by Kitchenham et al. [56] allowed us to develop and validate a strategy that revolved around specific research questions. Before stating these questions, an exploratory review was conducted to identify possible gaps in the subject of computational thinking and K–12 or K–9 education. Consequently, as part of our plan review phase, the research questions, search engine (database), keywords, and inclusion and exclusion criteria for document selection were stated. As suggested by the authors, the research questions were based on knowledge gaps identified in the field of study for a specific period [60,61].

4.1.1. Research Question and SLR Protocol

To generate the relevant information to answer the research question, three questions were established:

RQ1. Which countries incorporate Scratch in the formative plans for teachers in K–12 or K–9 education as a means to develop computational thinking (CT)?
RQ2. Which features of Scratch assist in teaching CT in K–12 and K–9 education?
RQ3. How does Scratch encourage the design of learning environments and teaching resources in K–12 and K–9 education?

4.1.2. Database and Search Terms

To include academic publications for this review, we selected the *ProQuest Central* database. We created a search string for each of the proposed questions (see Table 3). The three major terms were "Computational Thinking," "Scratch," and "K–12 or K–9 Education." The search strings were constructed using a Boolean "AND" to join the main terms and "OR" to include synonyms [31].

Table 3. Search strings in ProQuest Central.

No.	Research Question	Keywords	Results
RQ1	(AB(Scratch) AND ((Computational Thinking OR CT) OR (Programming OR Computer) Logic) AND (K–12 OR K–9) AND (Training OR Development OR Teaching OR Skills)) AND (stype.exact("Scholarly Journals") AND pd(20100426-20200426))	Scratch, Computational Thinking, CT, Programming Logic, Computer Logic, K–12, K–9, Training, Development, Teaching, Skills	38
RQ2	(AB(Scratch) AND ((Computational Thinking OR CT) OR (Programming OR Computer) Logic) AND (K–12 OR K–9) AND (Teaching OR Skills)) AND (stype.exact("Scholarly Journals") AND pd(20100426-20200426))	Scratch, Computational Thinking, CT, Programming Logic, Computer Logic, K–12, K–9, Teaching, Skills	36
RQ3	(AB(Scratch) AND (Learning Environment OR Educational Resources OR Facilities) AND (K–12 OR K–9) AND (Development OR Teaching OR Learning OR Learn OR Teach)) AND (stype.exact("Scholarly Journals") AND pd(20100426-20200426))	Scratch, Learning Environment, Educational Resources, Facilities, K–12, K–9, Training, Development, Teaching, Skills, Learning, Learn, Teach	37

4.1.3. Inclusion and Exclusion Criteria

Following Ramírez-Montoya's work [62], a set of detailed inclusion and exclusion criteria (see Table 4) was designed to identify whether a study could help answer the specified research questions [31]. The word Scratch by itself presented an issue, because the idiom "start from Scratch" is commonly used to refer to something or someone starting from the beginning or with no aid or help. We decided to include all studies that mentioned the word "scratch" in their "Abstract" sections. We did this intentionally to avoid any studies that mentioned Scratch merely as an example of a visual programming environment or as part of the idiom. Even with this condition, some studies mentioned Scratch simply as an example of a visual programming environment or idiom. The conduct review phase later addressed these studies by selecting only the ones that centered their research around the programming language Scratch. The scope of this review considered academic publications no older than ten years (2010).

Table 4. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Academic publications	Studies mention Scratch only as an example of a visual programming environment
Publishing date no older than 2010	Non-academic publications
Studies that focus on Scratch and computational thinking	The word "scratch" used as part of an idiom
Studies that center around K–12 or K–9 education	

4.1.4. Limitations of the Study

Possible limitations of this study include the use of ProQuest Central as the primary source used to conduct the study, and the use of additional databases to further complement the review for future works is encouraged. There are examples in the literature such Zhao's, which suggested that multiple database engines should be used for a valid systematic review. However, there is evidence offered by Bramer et al. regarding single database publications [63,64]. This is an ongoing debate among systematic literature review authors and a potential critique of this study, depending on the reader´s point of view. In light of Scratch's young age (launched in 2007), the review was limited to publications no older than a 10 years (2010–present). This fact creates a methodological debate for future work, we suggest that the date selection for articles be reviewed, given that programming tools, programming paradigms, and processing power are constantly evolving at a rapid pace.

Another limitation of this study was the primary focus on Scratch's impact on K–9 and K–12 education. There is no doubt that computational thinking extends beyond primary education, and thus future work should consider analyzing the impact tools like Scratch have on middle and higher education. Publication bias should be another factor to consider, as Kitchenham and Charters [65] stated that studies with positive results are more likely to be published than negative ones, resulting in an overly positive review of the use of Scratch rather than a full spectrum of views. In an effort to diminish validity threats posed to the study, we addressed each of the validity threats identified in Feldt and Magazinius's work [66]. We considered the external validity and transferability of our findings by stating the limitations of our work and presenting a general overview of the use of Scratch in the educational setting in different contexts. This was done to avoid wrongful generalization by only analyzing the use of Scratch in a particular setting. Regarding dependability, the methodology used for this study was stated in detail by presenting each of the steps taken to elaborate the report. The inclusion and exclusion criteria for document selection were included to enable the results to be consistent and replicable. As stated by Ramirez-Montoya, it is the nature of an SLR to introduce the findings of a specific subject from the existing literature. We are confident the reviewed publications are reliable sources of information, thus diminishing any credibility threats to our study [60].

4.2. Conduct Review

During this phase, studies that fulfilled the search terms for each research question were listed and input into a spreadsheet (link: https://figshare.com/s/3386a37cff72deee02ce (accessed on 4 January 2021). The search produced 111 results, which were then filtered to eliminate duplicates. Once the duplicates were removed, a detailed review was conducted to assess the relevance of the studies to the proposed research questions. This review discarded 81 studies as either duplicates or not having relevance to the research questions for this review, thus returning a total of 30 studies (see Table 5), that met the protocol developed in the first phase of the methodology. Figure 4 illustrates the process that took place for selection of the studies.

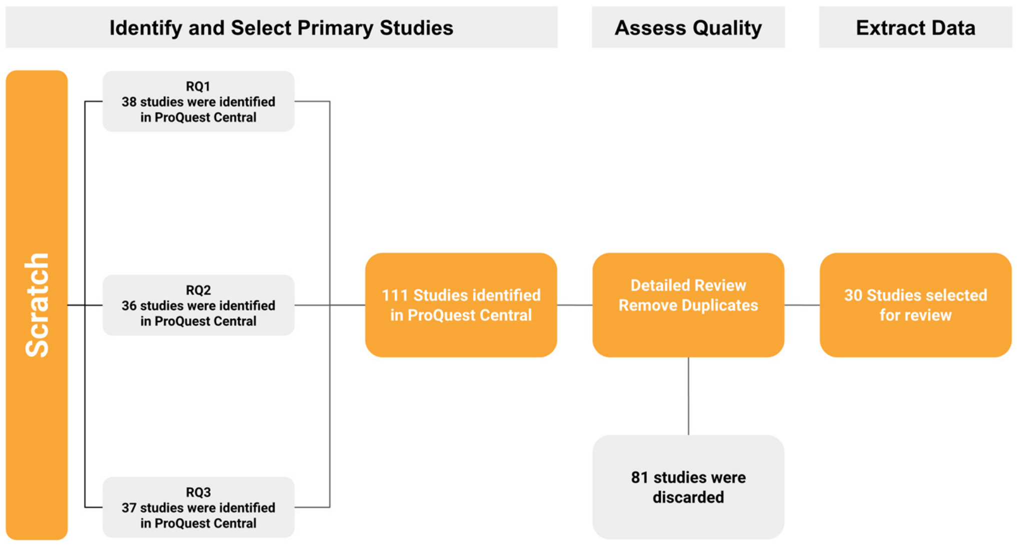
Identify and Select Primary Studies
Assess Quality
Extract Data
Scratch
RQ1 38 studies were identified in ProQuest Central
RQ2 36 studies were identified in ProQuest Central
RQ3 37 studies were identified in ProQuest Central
111 Studies identified in ProQuest Central
Detailed Review Remove Duplicates
81 studies were discarded
30 Studies selected for review

Figure 4. SLR conduct review phase.

Table 5. Studies selected for the review.

#	Authors	Item Type	Title
1	Halverson, E.; Sheridan, K.	Journal Article	The Maker Movement in Education
2	Burke, Q.	Book Section	DIY zones for Scratch design in class and club
3	Traylor, S.	Journal Article	Scratch that: MIT's Mitchel Resnick Says Kids Should Do It for Themselves
4	Demir, Ö.; Seferoglu, S.	Journal Article	Developing a Scratch-based coding achievement test
5	Moreno-León, J.; Robles, G.	Conference Paper	Code to learn with Scratch? A systematic literature review
6	Yadav, A.; Cooper, S.	Journal Article	Education fostering creativity through computing
7	Haduong, P.	Journal Article	"I like computers. I hate coding": a portrait of two teens' experiences
8	Gross, K; Gross, S	Journal Article	TRANSFORMATION: Constructivism, Design Thinking, and Elementary STEAM
9	Grover, S.; Jackiw, N.; Lundh, P.	Journal Article	Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school
10	Erümit, A.	Journal Article	Effects of different teaching approaches on programming skills
11	Deng, W. et al.	Journal Article	Pencil Code improves learners' computational thinking and computer learning attitude
12	Pellas, N.; Vosinakis, S.	Journal Article	The effect of simulation games on learning computer programming: A comparative study on high school students' learning performance by assessing computational problem-solving strategies
13	Oluk, A.; Korkmaz, Ö.; Oluk, H.	Journal Article	Effect of Scratch on 5th Graders' Algorithm Development and Computational Thinking Skills
14	Tang, K.; Chou, T.; Tsai, C.	Journal Article	A Content Analysis of Computational Thinking Research: An International Publication Trends and Research Typology
15	Yildiz Durak, H.	Journal Article	The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving
16	Hagge, J.	Journal Article	Coding to Create: A Subtext of Decisions as Early Adolescents Design Digital Media
17	Mladenović, M.; Boljat, I.; Žanko, Ž.	Journal Article	Comparing loops misconceptions in block-based and text-based programming languages at the K–12 level
18	Lee, Y.J.	Journal Article	Scratch: Multimedia Programming Environment for Young Gifted Learners
19	Grover, S.; Pea, R.; Cooper, S.	Journal Article	Designing for deeper learning in a blended computer science course for middle school students
20	Chang, C.	Journal Article	Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction
21	Sáez-López, J.; Cózar-Gutiérrez, R.	Journal Article	Programación visual por bloques en Educación Primaria: Aprendiendo y creando contenidos en Ciencias Sociales
22	Martin, C.	Journal Article	Libraries as Facilitators of Coding for All
23	Uzunboylu, H.; Kinik, E.; Kanbul, S.	Journal Article	An Analysis of Countries which have Integrated Coding into their Curricula and the Content Analysis of Academic Studies on Coding Training in Turkey
24	Lazarinis, F. et al.	Journal Article	A blended learning course for playfully teaching programming concepts to school teachers
25	Çakiroğlu, Ü. et al.	Journal Article	Exploring perceived cognitive load in learning programming via Scratch
26	Gabriele, L. et al.	Journal Article	Lesson Planning by Computational Thinking Skills in Italian Pre-Service Teachers
27	Adler, R.; Kim, H.	Journal Article	Enhancing future K–8 teachers' computational thinking skills through modeling and simulations
28	Romero, M.; Lepage, A.; Lille, B.	Journal Article	Computational thinking development through creative programming in higher education
29	Oluk, A.; Korkmaz, Ö.	Journal Article	Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables
30	Seiter, L.; Foreman, B.	Conference Paper	Modeling the learning progressions of computational thinking of primary grade students

5. Conclusions

Today, project-based research on developing a culture in informatics and a technical culture that includes professional development for primary teachers and people in charge of supervising teachers' actions is needed more than ever. The tools of ICT develop at an accelerated rate, creating a pressing need for curricular frameworks that can cope with their velocity. Research projects, like that of IE-CARE, must be set in motion to generate the knowledge required to rise to the challenges of whatever impact the pandemic might have on educational outcomes.

The theoretical contribution of this study is to provide elements to be considered in a teacher training framework to support learning computational thinking through Scratch in K–9 and K–12 education. Scratch is an innovative tool to help younger-aged students learn introductory programming concepts. The reviewed features of Scratch suggest that it not only has the potential to teach CT skills but also competencies in subjects that are not related to ICT. Its ease of use, accessibility, and trial-and-error approach make it an appealing tool for educators to implement, creating a bridge between the complex programming syntax and playful storytelling and animated scene creation. The ability to use Scratch for creativity and problem solving should inspire teachers to design and implement activities and resources that cater to students' academic needs.

This framework may also serve to support other disciplinary programs where technology plays an important role in the response to the COVID-19 educational disruption. A culture focused on the development of competencies among primary school teachers will prepare them for any future contingencies that disrupt or change education. An imperative need in our society is to prepare teachers with skills to design and deliver content efficiently, and this requires adequate experience with curricula and a reasonable time of practice. There are still significant issues to be addressed in the design and implementation of CT in educational curricula. Long gone are the days when research focused on whether technology in the educational setting would benefit the students. We need to move past that era and embrace the reality that technology is inherent in the learning processes and environments, especially in the context of educational disruptions like the one imposed by the COVID-19 pandemic. Future research should center on how and when technology should be incorporated into the classroom rather than discuss why and if it should be in educational settings. The COVID-19 disruption brought undesired challenges to the continuity of education. These must be addressed with the appropriate design and implementation of digital pedagogy and technological tools to ensure that the disruption in education is limited.

Author Contributions: Conceptualization, H.M. and M.G.G.-Z.; data curation, H.M. and M.G.G.-Z.; formal analysis, H.M. and M.G.G.-Z.; funding acquisition, H.M. and M.G.G.-Z.; investigation, H.M. and M.G.G.-Z.; methodology, H.M. and M.G.G.-Z.; project administration, H.M. and M.G.G.-Z.; resources, H.M. and M.G.G.-Z.; software, H.M. and M.G.G.-Z.; supervision, H.M. and M.G.G.-Z.; validation, H.M. and M.G.G.-Z.; visualization, H.M. and M.G.G.-Z.; writing—original draft, H.M. and M.G.G.-Z.; writing—review & editing, H.M. and M.G.G.-Z. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Authors can confirm that all relevant data are included in the article.

Acknowledgments: The authors acknowledge the contributions of Eric Brouillard and George-Luis Baron from Laboratoire EDA of the Université René Descartes, members of the IE-CARE project supported by the French National Research Agency (ANR). The authors would also like to acknowledge the financial support of Writing Lab, Institute for the Future of Education, Tecnologico de Monterrey, Mexico, in the production of this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Oluk, A.; Korkmaz, Ö.; Oluk, H.A. Effect of Scratch on 5th Graders' Algorithm Development and Computational Thinking Skills. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2018**, *9*, 1. [CrossRef]
2. Deng, W.; Pi, Z.; Lei, W.; Zhou, Q.; Zhang, W. Pencil Code improves learners' computational thinking and computer learning attitude. *Comput. Appl. Eng. Educ.* **2020**, *28*, 90–104. [CrossRef]
3. Hair, J.F.; Anderson, R.E.; Tatham, R.L.; Black, W.C. *Análisis Multivariante*; Prentice Hall: Madrid, Spain, 1999; ISBN 9788578110796.
4. Fagerlund, J.; Häkkinen, P.; Vesisenaho, M.; Viiri, J. Computational thinking in programming with Scratch in primary schools: A systematic review. *Comput. Appl. Eng. Educ.* **2021**, *29*, 12–28. [CrossRef]
5. Zhang, L.; Nouri, J. A systematic review of learning computational thinking through Scratch in K-9. *Comput. Educ.* **2019**, *141*, 103607. [CrossRef]
6. Moreno-Leon, J.; Robles, G. Code to learn with Scratch? A systematic literature review. In Proceedings of the 2016 IEEE Global Engineering Education Conference (EDUCON), Abu Dhabi, United Arab Emirates, 10–13 April 2016; pp. 150–156.
7. Simon, J.-C. *L'éducation et l'informatisation de La Société*. Documentation Francaise, 1980. Available online: https://www.epi.asso.fr/revue/histo/h80simon2.htm (accessed on 24 August 2020).

8. Baron, G.-L.; Drot-Delange, B.; Grandbastien, M.; Tort, F. Computer Science Education in French Secondary Schools. *ACM Trans. Comput. Educ.* **2014**, *14*, 1–27. [CrossRef]
9. Baron, G.-L.; Bruillard, E. Information technology, informatics and pre-service teacher training. *J. Comput. Assist. Learn.* **1994**, *10*, 2–13. [CrossRef]
10. Bruillard, E. Sesame Street et l'évaluation Des Technologies Éducatives. Available online: https://adjectif.net.shs.parisdescartes.fr/spip.php?article533 (accessed on 25 August 2020).
11. Wing, J.M. Computational thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
12. Yadav, A.; Hong, H.; Stephenson, C. Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends* **2016**, *60*, 565–568. [CrossRef]
13. Yadav, A.; Mayfield, C.; Zhou, N.; Hambrusch, S.; Korb, J.T. Computational Thinking in Elementary and Secondary Teacher Education. *ACM Trans. Comput. Educ.* **2014**, *14*, 1–16. [CrossRef]
14. Hsu, T.-C.; Chang, S.-C.; Hung, Y.-T. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Comput. Educ.* **2018**, *126*, 296–310. [CrossRef]
15. MIT Scratch—About. Available online: https://scratch.mit.edu/about/ (accessed on 6 September 2020).
16. Chang, C.-K. Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction. *J. Educ. Comput. Res.* **2014**, *51*, 185–204. [CrossRef]
17. Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E. The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* **2010**, *10*, 1–15. [CrossRef]
18. Baron, G.-L.; Voulgre, E. Initier à La Programmation Des Étudiants de Master de Sciences de l'éducation? {Un} Compte Rendu d'expérience. In Proceedings of the Sciences et Technologies de L'information et de la Communication en Milieu Éducatif: {Objets} et Méthodes D'enseignement et D'apprentissage, de la Maternelle à L'université, Patras, Greece, 24–26 October 2011.
19. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]
20. Salvo, M.J. *Constructionism in Practice: Designing, Thnking, and Learning in a Digital World—ProQuest*; Routledge: London, UK, 1998; Volume 7, ISBN 0805819843.
21. Dean, P.G.; Papert, S. Mindstorms: Children, Computers and Powerful Ideas. *Math. Gaz.* **1981**, *65*, 298. [CrossRef]
22. Kay, A. Squeak Etoys, Children & Learning. Available online: https://docs.huihoo.com/smalltalk/Squeak-Etoys-Children-and-Learning.pdf (accessed on 24 August 2020).
23. Steinmetz, J. Computers and squeak as environments for learning. In *Squeak: Open Personal Computing and Multimedia*; Guzdial, M., Rose, K., Eds.; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 2002; pp. 453–482.
24. Brennan, K.; Resnick, M. New Frameworks for Studying and Assessing the Development of Computational Thinking. In Proceedings of the annual American Educational Research Association meeting, Vancouver, BC, Canada, 16 April 2012; Volume 1, pp. 1–25.
25. Traylor, S. Scratch That: MIT's Mitchel Resnick Says Kids Should Do It for Themselves. *Technol. Learn.* **2008**, *29*, 27.
26. Csizmadia, A.; Curzon, P.; Dorling, M.; Humphreys, S.; Ng, T.; Selby, C.; Woollard, J. Computational Thinking A Guide for Teachers. Available online: http://computingatschool.org.uk/computationalthinking (accessed on 25 August 2020).
27. Robinson, K. *The Element: How Finding Your Passion Changes*; Ken, R., Lou, A., Eds.; Penguin Books: London, UK, 2009; ISBN 978-0143116738.
28. Segredo, E.; Miranda, G.; León, C. Hacia la educación del futuro: El pensamiento computacional como mecanismo de aprendizaje generativo. *Educ. Knowl. Soc. (EKS)* **2017**, *18*, 33. [CrossRef]
29. Troussas, C.; Krouska, A.; Sgouropoulou, C. Collaboration and fuzzy-modeled personalization for mobile game-based learning in higher education. *Comput. Educ.* **2020**, *144*, 103698. [CrossRef]
30. ANR. ANR Computer Sciences at School: Conceptualizations, Accompanying, Resources. Available online: https://anr.fr/Project-ANR-18-CE38-0008 (accessed on 31 December 2020).
31. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]
32. Gabriele, L.; Bertacchini, F.; Tavernise, A.; Vaca-Cárdenas, L.; Pantano, P.; Bilotta, E. Lesson Planning by Computational Thinking Skills in Italian Pre-service Teachers. *Informatics Educ.* **2019**, *18*, 69–104. [CrossRef]
33. Demir, Ö.; Seferoğlu, S.S. Developing a Scratch-based coding achievement test. *Inf. Learn. Sci.* **2019**, *120*, 383–406. [CrossRef]
34. Uzunboylu, H.; Kinik, E.; Kanbul, S. An Analysis of Countries Which Have Integrated Coding into Their Curricula and the Content Analysis of Academic Studies on Coding Training in Turkey. *TEM J.* **2017**, *6*, 783–791. [CrossRef]
35. Grover, S.; Pea, R.; Cooper, S. Designing for deeper learning in a blended computer science course for middle school students. *Comput. Sci. Educ.* **2015**, *25*, 199–237. [CrossRef]
36. Grover, S.; Jackiw, N.; Lundh, P. Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. *Comput. Sci. Educ.* **2019**, *29*, 106–135. [CrossRef]
37. MladenoviĆ, M.; Boljat, I.; Žanko, Ž. Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Educ. Inf. Technol.* **2018**, *23*, 1483–1500. [CrossRef]

38. Troussas, C.; Krouska, A.; Virvou, M.; Sougela, E. Using Hierarchical Modeling of Thinking Skills to Lead Students to Higher Order Cognition and Enhance Social E-Learning. In Proceedings of the 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA), Zakynthos, Greece, 23–25 July 2018; pp. 1–5. [CrossRef]
39. Lazarinis, F.; Karachristos, C.V.; Stavropoulos, E.C.; Verykios, V.S. A blended learning course for playfully teaching programming concepts to school teachers. *Educ. Inf. Technol.* **2018**, *24*, 1237–1249. [CrossRef]
40. Martin, C. Libraries as Facilitators of Coding for All. *Knowl. Quest* **2017**, *45*, 46–53.
41. Haduong, P. "I like Computers. I Hate Coding"': A Portrait of Two Teens' Experiences. *Inf. Learn. Sci.* **2019**, *120*, 349–365. [CrossRef]
42. Gross, K.; Gross, S. TRANSFORMATION: Constructivism, Design Thinking, and Elementary STEAM. *Art Educ.* **2016**, *69*, 36–43. [CrossRef]
43. Hagge, J. Coding to Create: A Subtext of Decisions as Early Adolescents Design Digital Media. *Technol. Knowl. Learn.* **2018**, *23*, 247–271. [CrossRef]
44. Yadav, A.; Cooper, S. Fostering creativity through computing. *Commun. ACM* **2017**, *60*, 31–33. [CrossRef]
45. Tang, K.-Y.; Chou, T.-L.; Tsai, C.-C. A Content Analysis of Computational Thinking Research: An International Publication Trends and Research Typology. *Asia-Pacific Educ. Res.* **2019**, *29*, 9–19. [CrossRef]
46. Çakiroğlua, Ü.; Suiçmez, S.S.; Kurtoğlu, Y.B.; Sari, A.; Yildiz, S.; Öztürk, M. Exploring perceived cognitive load in learning programming via Scratch. *Res. Learn. Technol.* **2018**, *26*, 26. [CrossRef]
47. Burke, Q. DIY zones for Scratch design in classand club. In *Creating the Coding Generation in Primary Schools*; Routledge India: New Delhi, India, 2017; pp. 81–100.
48. Halverson, E.R.; Sheridan, K. The Maker Movement in Education. *Harv. Educ. Rev.* **2014**, *84*, 495–504. [CrossRef]
49. Romero, M.; Lepage, A.; Lille, B. Computational thinking development through creative programming in higher education. *Int. J. Educ. Technol. High. Educ.* **2017**, *14*, 42. [CrossRef]
50. Pellas, N.; Vosinakis, S. The effect of simulation games on learning computer programming: A comparative study on high school students' learning performance by assessing computational problem-solving strategies. *Educ. Inf. Technol.* **2018**, *23*, 2423–2452. [CrossRef]
51. Lee, Y.-J. Scratch: Multimedia Programming Environment for Young Gifted Learners. *Gift. Child. Today* **2011**, *34*, 26–31. [CrossRef]
52. Erümit, A.K. Effects of different teaching approaches on programming skills. *Educ. Inf. Technol.* **2020**, *25*, 1013–1037. [CrossRef]
53. Oluk, A.; Korkmaz, Ö. Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. *Int. J. Mod. Educ. Comput. Sci.* **2016**, *8*, 1–7. [CrossRef]
54. Durak, H.Y. The Effects of Using Different Tools in Programming Teaching of Secondary School Students on Engagement, Computational Thinking and Reflective Thinking Skills for Problem Solving. *Technol. Knowl. Learn.* **2020**, *25*, 179–195. [CrossRef]
55. Seiter, L.; Foreman, B. Modeling the learning progressions of computational thinking of primary grade students. In Proceedings of the ninth annual international ACM conference on International computing education research—ICER '13, San Diego, CA, USA, 12–14 August 2013; pp. 59–66.
56. Adler, R.F.; Kim, H. Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Educ. Inf. Technol.* **2018**, *23*, 1501–1514. [CrossRef]
57. Arnedo-Moreno, J.; Garcia-Solorzano, D. Programming Is Fun! A Survey of the STEAM Digital Distribution Platform. In Proceedings of the 2020 IEEE 32nd Conference on Software Engineering Education and Training, CSEE and T, Munich, Germany, 1 November 2020; pp. 325–328.
58. López, J.M.S.; Gutiérrez, R.C. Programación visual por bloques en Educación Primaria: Aprendiendo y creando contenidos en Ciencias Sociales. *Revista Complutense de Educación* **2016**, *28*, 409–426. [CrossRef]
59. Kitchenham, B.; Pretorius, R.; Budgen, D.; Brereton, O.P.; Turner, M.; Niazi, M.; Linkman, S. Systematic literature reviews in software engineering—A tertiary study. *Inf. Softw. Technol.* **2010**, *52*, 792–805. [CrossRef]
60. Ramírez-Montoya, M.-S.; García-Peñalvo, F.-J. Co-creation and open innovation: Systematic literature review. *Comunicar* **2018**, *26*, 9–18. [CrossRef]
61. Ramirez-Montoya, M. Challenges for Open Education with Educational Innovation: A Systematic Literature Review. *Sustainability* **2020**, *12*, 7053. [CrossRef]
62. Ramírez-Montoya, M.-S.; Lugo-Ocando, J. Systematic review of mixed methods in the framework of educational innovation. *Comunity* **2020**, *28*, 9–20. [CrossRef]
63. Zhao, J.-G. Combination of multiple databases is necessary for a valid systematic review. *Int. Orthop.* **2014**, *38*, 2639. [CrossRef]
64. Bramer, W.M.; Rethlefsen, M.L.; Kleijnen, J.; Franco, O.H. Optimal database combinations for literature searches in systematic reviews: A prospective exploratory study. *Syst. Rev.* **2017**, *6*, 245. [CrossRef]
65. Kitchenham, B.; Kitchenham, B.; Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Available online: https://www.bibsonomy.org/bibtex/aed0229656ada843d3e3f24e5e5c9eb9 (accessed on 29 November 2020).
66. Feldt, R.; Magazinius, A. Validity Threats in Empirical Software Engineering Research—An Initial Survey. Available online: http://www.robertfeldt.net/publications/feldt_2010_validity_threats_in_ese_initial_survey.pdf (accessed on 25 April 2021).