sciendo

Research on Education and Media
Vol. 11, N. 2, Year 2019
ISSN: 2037-0830 – DOI: 10.2478/rem-2019-0023

Computational thinking in primary education: a systematic literature review

Panagiotis Kakavas[a], Francesco C. Ugolini[b]

[a]*Università Guglielmo Marconi, Roma, panoskakavas86@gmail.com*
[b]*Università Guglielmo Marconi, Roma, f.ugolini@unimarconi.it, https://orcid.org/0000-0002-7887-7983*

Abstract

This study presents a 13-year (2006–2018) systematic literature review related to the way that computational thinking (CT) has grown in elementary level education students (K-6) with the intention to: (a) present an overview of the educational context/setting where CT has been implemented, (b) identify the learning context that CT is used in education, (c) highlight the ways of assessment/measurement of CT and present the learning outcomes for students who engage in CT educational activities. A set of criteria were specified to select appropriate studies for inclusion in the review. A thorough search in ten large electronic databases, meeting the inclusion criteria, revealed 53 studies on CT in primary education. The results of the study revealed a variety of educational and learning contexts that CT has been integrated. The majority of studies use the framework of programming for both plugged and unplugged activities in order to cultivate students' CT-skills, while the main interest focuses on the subject of Computer Science and STEM field in general. However, teaching and learning issues on CT-concepts and skills, CT-measurement and the adoption of an established definition of CT remain a challenge. Based on the current findings, some recommendations and implications for future research are provided.

Keywords: *Computational thinking, primary education, systematic literature review, content analysis*

1. Introduction

1.1. Defining Computational Thinking

The concept behind Computational Thinking (CT) is not new; the best-known author is certainly Seymour Papert (1980) who fostered its development even in primary school children, although we have some precedents since the 1950s with the phrase 'algorithmic thinking'. But the importance of the current debate both from a scientific and a policy (EUN, 2015; MIUR, 2015) point of view actually originates from the seminal article by Jeannette Wing (2006). However, the development of CT in primary school is often confused with the coding process alone, which is actually the real programming activity, albeit using specific languages addressing younger people (the most famous is probably Scratch, developed by MIT). We may suppose that this is due to the prevalence in policy-making of the functionalistic paradigm (Dufva & Dufva, 2016), having the aim of promoting digital skills relevant to many career paths (European Commission, 2016).

But according to Wing's intent, CT is actually a thinking skill, which refers to the way a computer scientist acts in solving problems, rather than merely to the ability to program a computer: 'Conceptualizing, not programming. Computer science is not computer programming. Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction […]; a way that humans, not computers, think. Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative' (Wing, 2006, p. 35)[1].

[1] It is also true that in Wing's paper there were some naivety, especially when she claimed that CT involves 'understanding human behaviour, by

We have therefore to distinguish actions focused simply on computer science and programming constructs from the higher level thinking skills, more relevant from an educational perspective. Brennan and Resnick (2012), two of the Scratch authors, proposed an operational definition in which they discern 'CT Concepts' ('sequences, loops, parallelism, events, conditionals, operators, and data') from 'CT practices' that 'focus on the process of thinking and learning, moving beyond what you are learning to how you are learning' (Ivi, p. 7) ('being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing') and 'CT Perspectives' ('expressing, connecting, questioning').

Anderson (2016) makes a synthesis of several experiences, stating and formalizing a model in 5 major steps: 'decomposition, pattern recognition, abstraction, algorithmic design and evaluation'. This kind of definition relates CT mainly to a problem solving process, according to the Barr and Stephenson (2011) definition: 'CT is an approach to solving problems in a way that can be implemented with a computer' (Ivi, p. 115).

We must therefore give a different role to the 'testing and debugging' practice, that is considered in many definitions (apart the definition from Brennan and Resnick that we have recalled, see also Grover and Pea [2012]). While decomposition, pattern recognition, abstraction and algorithmic design are mostly related to the program design rather than coding in the strict sense, and could also be developed with unplugged activities, 'testing and debugging' relates to the actual programming phase, taking advantage of the features of the computer to give immediate feedback to the learners' actions, which can be useful in order to enhance self-regulatory skills (Trinchero, 2019).

Another approach that is worth mentioning from an educational perspective is the one that sees computing as a creative human activity (Grover, Pea, 2012; Verborgh, 2013; Olimpo, 2017). Brennan and Resnick (2012), within their 'expressing' CT perspective, state that 'a computational thinker sees computation as a medium and thinks, "I can create." and "I can express my ideas through this new medium".' (Ivi, p. 10). Dufva and Dufva (2016) link this issue to a specific paradigm of coding, the 'postmodern' one: 'Creative coding allows artists to question and critique code and, at the same time, express themselves through code' (Ivi, p. 105).

This is the overall perspective we intend to adopt in this systematic review: as many policies strongly promote CT in primary school, it cannot be reduced to simple computer science skills nor should it lead children to think like computers, but rather, it should help them to develop higher level thinking abilities in order to correctly interact with digital technologies.

1.2. A systematic review about Computational Thinking in primary education

When Jeannette Wing first introduced the term '*Computational Thinking*' in 2006, she also suggested that all the educational community had better contribute in students' CT–skill acquisition. Since then, there have been many initiatives and attempts to introduce children of all educational levels to CT-concepts, by cultivating their CT-competences during their learning of diverse subjects, not only STEM.

To this end, it would have a great interest to investigate the various teaching approaches that have been implemented in the elementary and the secondary educational level with the intention to help students develop or assess various CT-skills. Therefore, the aim of this paper is to review the CT-literature from 2006 to 2018 related to primary education (K-6) – Kindergarten to 6th grade – in order to examine the ways that CT has grown and assessed in elementary level education students. For the needs of this study, the Greek educational system was taken into account. Thus, Kindergarten-students are considered the children aged 4 to 6 years old, and Primary-students, the children attending the grades from 1st to 6th (age level: 6–12 years old).

Although we could not find studies with the same goals, some reviews with similarities have been reported. The systematic literacy review of Lockwood and Mooney (2018) has as its objective to open up a space for secondary-level educators that are interested in implementing CT into their instruction by providing them with classroom choices and ideas on how to succeed in this direction, as well as education researchers are informed through a detailed overview of what work has been conducted in the field, as well as bringing the focus in some gaps and potential opportunities for further work that exist. Moreover, Araujo, Andrade and Guerrero (2016) focused on the identification and the classification of the approaches so as to spread the CT as well as on the various ways of evaluating CT competences, while Shute, Sun and Asbell-Clarke (2017) examined the expanding domain of CT within the educational field, indicating the existing diversity in the way that it is defined and evaluated as well as the different models and interventions that have been noted. Furthermore, Flórez, Casallas, Hernández, Reyes, Restrepo, and Danies (2017) make an analysis, as well as a discussion of the outcomes of their conducted and reviewed study-cases, pinpointing the significance of 'learning programming' with the main focus being the nurture of the CT-skills in children starting at a young age.

In addition, the study by Hava and Cakir (2017) makes a presentation of the outcomes of a systematic literature review on the development of the educational computer game and its implementation in learning contexts by researching the impact of the game design task on the 'learning outcomes' of the students. Moreover, Ioannou and Makridou (2018) made

drawing on the concepts fundamental to computer science' (*Ivi*, p. 33), while it is now well-known that humans and machines do not think in the same way thanks to the embodied cognition and situatedness theories (Varela, 1990; Gallese, Lakoff, 2005; Berthoz, 2009; Rivoltella 2012; Sibilio; 2012; Rivoltella, Rossi, 2019).

a review of the published literature referring to the intersection of CT and educational robotics, specifically with the main focus being the exploitation of educational robotics in developing the CT-skills of the learners in K-12. Finally, Dagienė and Stupurienė (2016) objective was the introduction of the Bebras model in cultivating CT, implementing a ten-year observation and contest in diverse countries, while Ching, Hsu and Baldwin (2018) provided an overview of the opportunities for CT development in young students that go beyond the conventional computers and computing devices and alternatively, make use of toys and board games.

Despite the above, a review study focusing on the didactic approaches for the cultivation of CT-skills by primary level education students has not yet been reported. Thus, this systematic review would help us to highlight teaching and learning approaches, as well as ways of evaluation that have been used to cultivate and assess students' CT-skills. Additionally, the review would help other researchers to devise new CT teaching frameworks/approaches and ways of its assessment by implementing them upon the students of primary education even in disciplines not related to STEM fields.

The rest of this paper is organized as follows: In Section 2, the research methodology is reported. In Section 3, the categories of the reviewed papers along the examined dimensions are depicted. The results of the study are presented in Section 4, followed by their conclusions of the review in Section 5.

Deeper methodological details are presented in the Annexes attached with the main paper.

2. Methodology – Context of the study

2.1. Research question

The main question of this review paper is its attempt to answer the way that 'computational thinking' has evolved in elementary level education students (K-6). Therefore, the objective of the paper is to review the literature from 2006 to 2018 on the development of 'computational thinking' in the primary level of education, with the intent being:

(a) present an overview at the context/settings of education where CT has been implemented,
(b) identify the learning context that CT is used in education,
(c) highlight the ways in which CT has been evaluated and present the learning outcomes for students who engage in CT educational activities.

2.2. Data collection

2.2.1. Databases searched

This study reviews papers published in journals of science, international-conferences proceedings, workshops and symposiums since the beginning of January 2006 up to the end of December 2018.

To achieve this, ten big electronic databases that are related to 'educational contexts', 'digital technology' and 'social science' were being used in the review in focus. More specifically, it makes use of: *SpringerLink*, *ACM* (Association for Computing Machinery), *Bio-Medical Library*, *ERIC* (Education Resources Information Centre), *IEEE Xplore Digital Library*, *Taylor & Francis Online*, *Wiley*, *LearnTechLib* (Learning & Technology Library), *Ingenta Connect* and *Science Direct*.

2.2.2. Search term

The search was conducted using the keywords 'computational thinking'. The search was limited to the period from January 2006 to December 2018. As a result, 3,547 papers/items were identified.

2.2.3. Selection of papers to be included in the review

The papers selected in the review were carefully chosen under specific criteria. Through a full screening of the identified papers and a close look at each paper's title, abstract and content, as well as an exclusion of those papers that were not relevant to the inclusion criteria, 53 papers were piled up. The inclusion criteria so as for the papers to be included in the review are reported below. The academic papers had to:

(a) make an explicit reference to the term 'computational thinking' in the *title* and/or *abstract* and/or *keywords*,
(b) be written in *English language*,
(c) have a typical form of a *scientific paper* (no posters, roundtables, work in progress papers, short papers, etc.),
(d) focus on *CT and primary level education students* (K-6),
(e) present *empirical data* – emerged from methodologically sound empirical studies – either collected by kindergarten students or by K-6 elementary students, by providing adequate information about the *research methodology*, the *participants* and the *research procedure used* (no pilot and preliminary studies, exploratory studies).

Table 1 shows the number of papers found from each database as well as the number of academic papers that eventually constituted the body of this review after their inclusion in it.

Table 1. Number of papers/items emerged from each database and number of included papers in the review

Databases Searched	Number of Papers/items Identified in Search	Number of Papers Meeting the Inclusion Criteria
Springer	1,118	9
ACM	753	15
Bio-Medical Library	17	1
ERIC	192	3
IEEE Xplore Digital Library	325	4
Taylor and Francis Online	273	8
Wiley	172	2
LearnTechLib	393	6
Ingenta Connect	65	-
Science Direct	239	5
Total	3,547	53

Finally, using 'the above mentioned criteria', from the 3,547 papers, a total of 53 papers met the criteria so as to be included in the review and were evaluated as 'relevant' for full text review. Twenty-eight of them are published *in 'scientific journals'*, sixteen at *'international conferences'*, seven are presented at *'international symposiums'* and two at *'international workshops'*.

The twenty-eight 'journal papers' found were published in twenty-two diverse journals. All papers were published in computer, education and related to technology journals. All journals are abstracted/indexed in various databases such as: Scopus, ProQuest, EBSCO, INSPEC, ERIC, Google Scholar, PsycINFO, Gale, ERA, SCImago and ERIH PLUS. See Annex A for further details about Data Collection.

The distribution of the reviewed papers by publication year is indicated in Fig. 1.

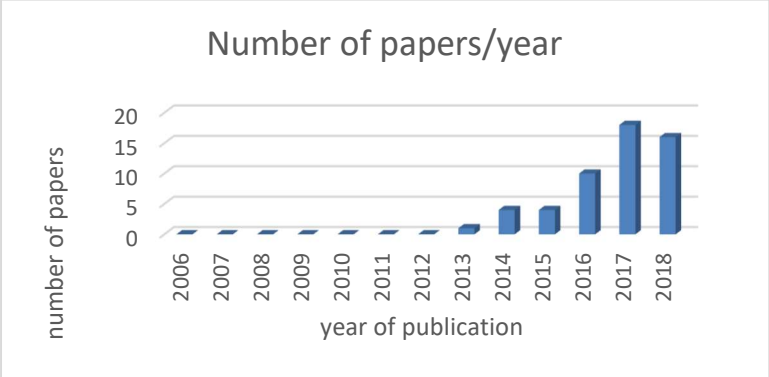

Fig. 1. Distribution of the reviewed papers by publication year

It is evident that in the last five years – and specifically in 2017–2018 – there is an augmenting interest of the scientific educational community for the cultivation of CT-skills to primary school students.

3. Data analysis – Coding of papers

The reviewed papers that met the inclusion criteria were coded through a proforma data extraction that were created after taking into consideration the aforementioned 'research questions', which categorized the reviewed papers along some important dimensions. Therefore, the papers were categorized in relation to:

(a) *context/settings* of education that CT has been implemented in primary education (educational level, location of the study, learning subject, relation to STEM fields, duration of the study),

(b) *learning context* that CT has been incorporated in primary education (main CT-concepts approached, way of integration, digital environments used, type of activities, use of programming language, type of programming language used),

(c) the *ways of 'assessment or measurement' of CT* and '*learning outcomes'* for students who engage in CT educational activities (sample used, location of the study, data collection tools, outcomes of the studies).

4. Results

4.1. Educational context of CT implementation

Mainly, the results show that the *educational level* of the conducted studies do not only concern the primary school students (31 papers) but also the students attending Primary–Middle level of education (14 papers). As for the *location* of these studies, most of them were conducted in the school settings (32 papers), such as the school classroom or the school's computer lab. The reviewed studies focused on various *learning subjects* but the great majority of studies (44 papers) focused on STEM disciplines for developing CT-competences to primary school students. As for the duration of the studies, most of them lasted for up to one week (18 papers), while about equal number of papers (17) show that the duration of the studies varies between one to six months.

Table 2 shows a synthesis of the results about the educational context; a deeper analysis and further details can be found in Annex B.

Table 2. Educational context of CT implementation (synthesis)

Educational Level	Location of the Study	Learning Subject	Duration of the Study
Primary: 31	School: 32	Computer Science: 33	Until 1 week: 18
Primary–Middle: 14	Out-of-school: 13	Other STEM: 11	More than 1 week to 1 month: 7
Primary–Middle–High: 4	Online: 1	Non STEM: 9	More than 1 month to 6 months: 17
Kindergarten–Primary: 3	More than one setting: 4		More than 6 months: 7
Kindergarten: 1	Not mentioned: 3		Not mentioned: 4
TOTAL: 53			

4.2. Learning context of CT incorporation

The ways through which CT has been integrated in teaching practices for students' CT-skills development refer mainly to Plugged Programming Activities (15 papers), Game Programming & Game Activities (10 papers) and Robotics (7 papers). Most of the interventions conducted were based on plugged activities (34 papers) for students' CT-development. Unplugged activities were also conducted to a lesser extent (10 papers), while there were efforts to combine plugged and unplugged activities for the same purpose (8 papers).

Papers proposing plugged activities make use of one or more than one digital environment: Scratch is by far the most popular (16 papers), followed by Alice (4 papers), Kodu (3 papers) and Agentsheets (3 papers). 19 other different digital environments occur once or twice in 23 papers. 38 studies used a programming digital environment for students' CT-nurture with the block-based visual programming languages, which constitutes the majority of the programming environments used (36 out of 38).

The review study obtained that the concepts of abstraction (41 papers), algorithms and procedures (41 papers) and control structure (33 papers) constitute the most common CT-concepts and competences approached in primary education. Moreover, the concepts of testing and verification (19 papers), problem decomposition (18 papers), parallelization (16 papers) and data analysis (13 papers) were approached to a lesser extent followed by the concepts of simulation (8 papers), data representation and analysis (8 papers), data collection (6 papers) and automation (2 papers).

Table 3 shows a synthesis of the results about the learning context of CT incorporation; a deeper analysis and further details can be found in Annex C.

Table 3. Educational context of CT implementation (synthesis)

Way of Integration	Type of Activities	Use of Programming Language
Plugged programming activities: 15	Plugged: 34	Yes: 38
Game programming and game activities: 10	Unplugged: 10	No: 14
Robotics: 7	Combination: 8	Not mentioned: 1
No CS education activities: 6	Not mentioned: 1	
Unplugged programming activities: 4		
Plugged and unplugged programming activities: 4		
Paper activities: 4		
Simulation activities: 3		
TOTAL: 53		

4.3. CT evaluation

The reviewed empirical studies referring to primary level of education students are categorized below in terms of: (a) *'context of the assessment'*, and (b) *'outcomes of the studies'*.

As for the *'context of the evaluation'*, it is significant to be noted that there were few studies aiming not only to state hypotheses but test them as well making use of empirical studies using *'experimental research methods'* (Cohen, Manion & Morrison, 2007). Moreover, various *'tools for data collection'* were used, with pre/post assessments (28 papers), artefacts' analysis (17 papers), questionnaires (17 papers), and interviews/discussions (15 papers) being among the most common ones. Besides, a combination of different kind of such tools was used in the majority of the studies so that the results are more valid and reliable.

Table 4 shows a synthesis of the results about the learning context of CT assessment; a deeper analysis and further details can be found in Annex D.

Table 4. Context of the CT-assessment of the empirical studies

Sample Size: Number of Participants	Tools for Data Collection (may be more than one)
Up to 10: 4	Pre/Post assessments: 28
11–30: 7	Artefacts' analysis: 17
31–50: 9	Questionnaires: 17
51–100: 15	Interviews/discussions: 15
More than 100: 16	Observations: 10
Not mentioned: 2	Log-files: 8
	Pre/Post surveys: 7
	Rubrics: 6
	Various recordings (video, audio, screen-recordings, photographs): 5
TOTAL 53	

In addition, the analysis of the main *'outcomes on CT determined in the reviewed papers'* formed around 13 themes. However, it was observed that the emerging findings had 'positive results' in the context of CT in general. Table 5 lists these main themes: a deeper analysis and further details can be found in Annex E.

Table 5. Main findings emerged from the reviewed studies for CT implementation in primary education

Main Findings on CT for Primary Education	*f*
1. Findings on CT-skills	42
2. Findings on students' attitudes	31
3. Findings on CT-position in curricula	24
4. Findings on other skills cultivated	20
5. Findings on behavioural issues	15
6. Findings on gender issues	8
7. Findings on CT-practices cultivated	7
8. Findings on factors affecting learning outcomes	7
9. Findings on evaluation of CT	6
10. Findings on domain knowledge beyond CT and CS	5
11. Findings on differences with traditional educational methods	3
12. Findings on differences between students' age	2
13. Findings on awareness of computing	2

4.4. Operational Definitions and Assessment Tools

Analysing the papers included in the review, we can actually confirm that there is no commonly accepted and adopted definition of CT. Moreover, there is still a lack of assessment tools able to effectively measure CT and its development through didactic intervention, and this can limit the spread of CT programs in K-6 education (Grover, 2015). We therefore intend to extract from this systematic review some useful hints regarding the definitions most commonly adopted and the potential assessment tool being used in research.

To this extent, we propose here a more profound analysis of the 18 papers with both pre- and post-assessment (even if they cannot all be considered experimental, which would also mean having a control group). Moreover, in three of them, CT is the Independent Variable rather than the Dependent Variable (for instance, [49][2] aims to show to what extent CT programs increase Critical Thinking, Creative Thinking and Problem Solving). In the following sub-paragraphs, we therefore consider only those 15 papers with pre- and post-assessment related to CT.

[2] We refer to the reviewed papers by their ID number between brackets. For the full list of the reviewed papers, see Annex F.

In Table 6, we can see the distribution of the Operational Definitions adopted in the 15 papers we considered.

Table 6. CT Operational Definitions

CT Operational Definition	Number of papers
Brennan & Resnick (2012) - Explicit	2
Brennan & Resnick (2012) - Implicit	2
Barr & Stephenson (2011)	3
CSTA-ISTE (2011)	2
'4 steps' (Anderson, 2016)	1
Grover & Pea (2012)	1
CSLT - Implicit	1
No identifiable definition	3
Total	15

Several papers[5,21,31,36] adopt the Brennan & Resnick (2012) definition referred to in the introduction to this paper (see Fig. 2). Karen Brennan and Mitchel Resnick are two of the authors of Scratch, the most famous coding environment for children. According to our approach, this definition has the merit of clearly distinguishing the CT Concepts mostly related to Computer Science ('Sequences, Loops, Events, Parallelism, Conditionals, Operators, Data'), from the CT Practices ('Being incremental and iterative, Testing and Debugging, Reusing and Remixing, Abstracting and Modularizing') and the CT Perspectives ('Expressing, Connecting, Questioning').

Not surprisingly, this definition is referred to mainly in the papers reporting research with Scratch tools. But while CT Concepts are often measured by tests, in our review, we did not find an assessment tool focusing strictly on Brennan and Resnick CT Practices and CT Perspectives. In some cases, we did find items in wider tests focusing on some specific aspects according to the research aims.

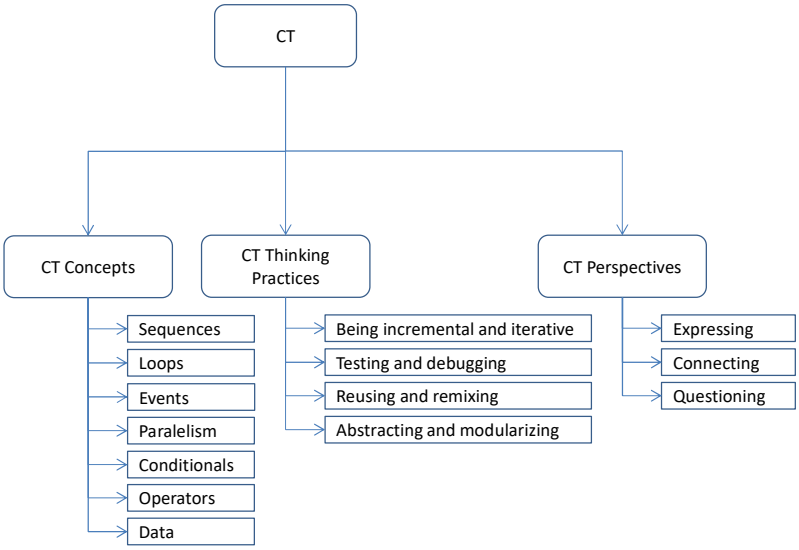

Fig. 2. Brennan and Resnick (2012) – Operational Definition

Three papers[28,47,48] adopt the basic definition stated by Barr and Stephenson (2011): CT is 'an approach to solving problems in a way that can be solved by a computer' (*Ivi*, p. 115). Two of them are ascribable to the same research group, and especially highlight the property of this problem solving methodology to be transferred and applied across subjects. This is clearly not an operational definition, but in any case it addresses a scenario-based assessment tool aiming to verify if CT Concepts can be applied in a different situation than the computing environment in which they were developed. Two more papers[9,29] refer to the Operational Definition stated by the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) (2011); this definition is actually an extension of Barr and Stephenson's (actually Barr and Stephenson report the building process of the CSTA-ISTE definition). It also focuses on 'Data analysis, organization and representation', 'Problem decomposition', 'Abstraction', 'Algorithmic thinking', 'Automation', 'Simulation' and 'Parallelization', all properties relating more or less directly to a problem solving process. The CSTA-ISTE definition could therefore be a good representation of the CT features addressing higher level thinking skills, but it is limited by the absence of a stable model: the original CSTA-ISTE statement refers to 'a problem-solving

process that includes (but is not limited to)' some characteristics, so that both papers need to do further elaboration in order to build their assessment tool, which cannot therefore be taken as a common reference.

One paper[7] adopts a definition built upon 4 progressive steps: Decomposition, Pattern recognition, Abstraction, Algorithmic design, recalling the design process used by software engineers. We can find this definition also in papers other than the 15 that we are examining here. It also somehow refers to a problem solving process that follows the Barr and Stephenson principle, but focuses more on the description of a top down approach rather than the specific elements of a thinking skill. Another research[3] adopts a similar definition, focusing on a 'thought process' that utilizes the elements of 'Abstraction', 'Generalization', 'Decomposition', 'Algorithmic thinking' and 'Debugging', referring to Grover & Pea (2013) and Selby & Woollard (2013).

One paper[51] deserves a specific mention: its CT Assessment tool involves the Computational Thinking Levels Scale (CTLS), referring implicitly to an Operational Definition, which includes a creative dimension; therefore, it is somehow related to the approach recalled in the introduction that sees computing as a creative human activity. We will describe the Assessment tool in more detail below.

Finally, three papers[4,19,52] do not declare their definition explicitly, nor can it be deducted from the Assessment Tool, as this one mainly focuses only on CT Concepts.

In Table 7, we can see the distribution of the Assessment Tools adopted in the 15 papers we consider.

Table 7. CT Assessment Tools

CT Assessment Tools	Number of papers
CT Concepts	4
Self-developed Test/Scale	6
Specific Embedded Test	2
Román-Gonzalez CT Test	2
CSLT Scale	1
Total	15

With regard to the assessment tools, the situation we found from our analysis appears to be even worse, compared with our aims: while looking for tests or scales that 1) focus on CT Practices or Perspectives rather than only CT or CS Concepts; 2) are easily re-usable and 3) are valid and reliable to some extent, we are led to the conclusion that it is only 3 out of 15 papers that meet the aforementioned conditions.

We can first exclude 4 papers that actually restrict themselves only to measuring the learning of CT or CS Concepts.

Moreover, we should not consider a substantial number of them (6 papers out of 15), whose authors developed their own test or scale according to their specific research aims. This is not fair for us for two reasons: first, from a metric point of view, such tests or scales did not face any validity or reliability check; second, from a theoretical perspective, they are strongly affected by the specificity of the research, their aims, and also their CT Operational Definition (the 6 papers refer to 4 of the aforementioned Operational Definitions).

Two more papers[5,52] embedded the assessment tools within the programming environment they used in the CT Developing Program, and are also very specific, though otherwise they are of some interest. They can both be connected to the same research group, which uses an Open Ended Learning Environment (OELE) focusing on the ability of learners to represent knowledge in models. The assessment tool is based on the comparison of the learners' model with the expert one, and therefore, the CT Vital Practices are somehow indirectly measured. The instrument may be of some interest, but the specificity of the environment makes it hardly transferable to other situations.

Probably, the most interesting assessment tool from our perspective is the CT Test developed by Marcos Román-Gonzalez and adopted by two papers[7,31] involving Román-Gonzalez himself. To the extent of our review, this test can go beyond the label of a self-developed test, as it has passed a rigorous validation process, involving content validity (Román-González, 2015), criterion validity (Román-González, Pérez-González, & Jiménez-Fernández, 2017) and convergent validity (Román-González, Moreno-León, & Robles, 2017). The CT Test is composed of 28 multiple choice items, executed on Google Forms technology, and it recalls an OCSE-PISA test as each item presents a problem that can be solved activating higher level cognitive processes. In this case, the respondent has to activate to a greater or lesser extent the four main cognitive processes related to CT: 'Decomposition', 'Pattern Recognition', 'Abstraction' and 'Algorithmic Design'. For this reason, we consider it not to be comparable to a mere CT Concepts Test.

Finally, we mention a Turkish research,[51] which adopts a particular approach. CT in 5th grade students is measured by a self-perception scale inside a wider test, called Computational Thinking Level Scale (CTLS), first conceived to address higher education students (Korkmaz, Çakır and Özden, 2017) and then adapted for middle school students (Korkmaz, Çakır and Özden, 2015). The scale is composed of 22 items (five points Likert type), grouped in 5 factors. The authors of the scale seems to trace CT back to a broader problem solving process (Korkmaz and Bai, 2019), as only one of these factors explicitly refers to computation ('Algorithmic Thinking'), while the others correspond to more general

cognitive processes ('Creativity', 'Critical Thinking', 'Cooperation', 'Problem solving'). However, they pay much attention to metric issues, and the scales appear to be transferable and reliable.

In conclusion, finding out a unified definition of CT is still a challenge; while many papers outline it as a Problem Solving thought process, they often restrict it to the CT Concepts while designing empirical research. When focusing on the Assessment Tools, the scenario appears to be even worse, as there is no validated tool commonly adopted addressing CT at a deeper level. In the papers we analysed, the only convincing solution we could find is the scenario-based or problem-based one, as the answerer should activate the typical cognitive processes usually adopted by computer scientists when designing software, even if, ultimately, they did not ignore the CT Concepts, and the Román-Gonzalez CT Test is currently the only one adopting this approach that is also transferable and validated.

5. Discussion

The findings of the aforementioned categories are discussed below in order for the current trends on CT to be explained and future studies in primary education to be followed. Moreover, the discussion followed by some recommendations according to the research gaps emerged in the literature. Finally, the limitations of this study are also discussed.

5.1. Main findings of the study

From the included papers analysed in this review, the following findings regarding CT's incorporation in the primary level of education are of significance:

- *It is evident that throughout the last five years, there is an increasing interest of the scientific educational community towards the development of CT-skills of primary school students.* Thus, we could assume that studies focusing on CT-cultivation of primary school students will increase in the years ahead.
- *Majority of the studies focus on STEM disciplines and specifically on the subject of CS, robotics and science*. Therefore, students' CT-skills cultivation through other disciplines beyond STEM field remains a challenge.
- *Most studies use the computer as a means of CT-cultivation*. Therefore, unplugged activities (e.g., Bebras tasks, puzzle-solving activities) are used to a lesser extent and research data from such studies are limited.
- *Majority of the studies use programming as a context for CT-development*. Of course, *programming* is a framework of CS and CT practice, and thus, the exclusive focus on programming for the development of CT is not only a pedagogical but also a methodological error, since the focus should be on higher-level concepts that should be taught, as well as on multiple cognitive domains/disciplines to which they should be applied (Voogt et al., 2015).
- *Most studies attempt to cultivate students' CT-competences through diverse activities that require the use of visual programming languages with Scratch, Alice and Kodu being the most common ones*.
- *It is also observed that most of the studies have been applied to the students attending the upper grades of elementary school*. Thus, more emphasis should be given to students attending lower grades even to Kindergarten students, so that we have a more comprehensive view on how CT could be integrated in primary education in general.
- *Majority of the studies focus on students' CT-skills development by highlighting and examining their artefacts, without, however, studying the influence of specific independent variables – such as the influence of previous programming experience, gender, age, pair programming – on the cultivation of CT sub-skills, practices or even perspectives.*
- *CT-measurement remains a challenge and an open-ended issue*. So, the field of CT requires systematic evaluation procedures (Lee et al., 2011) so as to reliably measure the different aspects of CT that will overcome the assessment of simple and local programming constructs (Armoni, 2016).
- *Most studies focus on CT-skills cultivation in a short-term period, while overlooking the long-term effects that CT has on students' career*. No studies have been undertaken examining whether the teaching/learning of CT has a direct or indirect impact on students' academic performance, career, faculty choice, or even long-term problem-solving skills (Lockwood and Mooney, 2018).
- *Teaching and learning issues on CT concepts and skills still remain open*. Due to the research field of CT being in its infant stages, the below open-ended queries about CT-teaching and learning formulated several years ago by other researchers (Wing, 2008; Barr & Stephenson, 2011) should be answered. The most important of them regard: (a) the concepts and competences that students can best learn at each grade in primary school, as well as (b) the effective sequencing of concepts in teaching children as their learning capability evolves over the years.
  See Annex E for further details.

5.2. Recommendations for future research

After the analysis of the main findings of the reviewed studies, significant research gaps in the literature emerged. Therefore, future research in the field could make a contribution to the existing literature. What follows is suggestions for

further research in CT that will be carried out in the primary education level. In particular, it is proposed that future studies should:

- Take advantage of unplugged activities to further detect the impact of these activities on CT-cultivation, as well as to examine the way of transition from such activities to plugged activities.
- Not only include programming activities, but also activities aimed at cultivating CT through other subjects, beyond STEM field.
- Make use of 'text-based programming languages' that are popular nowadays, such as Python, JAVA, C^{++} and PHP. Important findings could emerge by comparing the use of these languages in teaching practice, as well as with the various visual programming languages for students' CT-development.
- Examine specific factors (e.g., gender, age, programming or STEM previous experience) that could affect concrete CT-skills, practices or even perspectives.
- Focus on the creation of more 'valid' and 'reliable' assessment tools and specifically on the development of general CT assessment instruments of each students' age that could evaluate students' CT-skills in a wider range of activities.
- Investigate the long-term effect of CT in areas related to students' academic progress and career, problem solving skills, the impact on their performance in other subjects, as well as the impact of CT ability on students' daily life.
- Examine the relationship between CT and other 21st century thinking skills such as problem solving, creative thinking, critical thinking, metacognitive skills and so on.
- Classify the concepts/skills that students can best learn at each grade by proposing an 'effective ordering of concepts' in teaching students as their learning ability growths over the years.
- Attempt to set concrete cognitive goals for each grade by providing specific paradigms and activities, as well as guidelines for teachers to follow while incorporating CT in their teaching practice.
- Conduct and provide discussions and findings from similar and periodic reviews so that the field is constantly evaluated, and future guidelines are designed.

Future studies that will be conducted taking into account the aforementioned recommendations could contribute to the literature in order for CT to be integrated in curricula in an effective way, and thus, to form a structure – commonly accepted – that CT could be implemented in diverse aspects of everyday life by the whole literate population.

5.3. Limitations of the study

The current study has a number of limitations, since it was restricted by the specific search term used, the period in which the reviewed studies were published, as well as the scientific databases searched. However, the findings emerging from the current study provide a useful overview of the research in the way that CT has been incorporated and evaluated in primary education, which is representative of the situation so far.

6. Conclusions

The aim of this paper was to make available a snapshot of the current research and work around the way that CT has grown in elementary level education students. For this purpose, a systematic literature review was conducted from 2006 to 2018 based on 53 papers, found 'relevant for inclusion', after searching ten large electronic databases by using precise keywords. To this end, the research questions made in this paper were chosen with the intention to provide an overview to researchers and educators of how CT could be better embedded into the school classroom.

The findings revealed that the majority of the studies use the framework of programming for both plugged and unplugged activities in order to cultivate students' CT-skills. It was also confirmed that most studies focus on the subject of CS and STEM field in general. Of course, there are various tools and ways proposed – digital being the most common – with the intention to incorporate CT in teaching practice. It was also shown that the most frequent CT-concepts/skills approached regard '*abstraction*, *algorithms and procedures*, *control structures*, *testing and verification*, *problem decomposition and parallelization'*. Moreover, it should be noted that further work should be done on the creation of more valid and reliable evaluation tools in which CT can be assessed.

It is hoped that this literature review be useful for future policy makers, curriculum designers, researchers and teachers who desire to implement CT into the teaching practice, as well as form a basis and a consensus on how CT could be taught in primary education. It would contribute to the effective teachers' education by providing them with necessary information about CT and giving specific guidelines and instructions on how to teach CT to the right students at the right age level.

Authors' contributions

Francesco C. Ugolini is the author of paragraphs 1.1 and 4.4. Panagiotis Kakavas is the author of the remaining of the paper.

References[3]

Anderson, N.D. (2016). A call for computational thinking in undergraduate psychology. *Psychology Learning & Teaching*, 15, 226–234.

Araujo, A.L.S., Andrade, W., & Guerreno, D. (2016). A Systematic Mapping Study on Assessing Computational Thinking Abilities. In *Proceedings of the 46th IEEE Frontiers in Education Conference, FIE '16*, (pp. 1–9). Bayfront Convention Center, Erie, PA, USA.

Armoni, M. (2016). Computer Science, Computational Thinking, Programming, Coding: The Anomalies of Transitivity in K-12 Computer Science Education. *ACM Inroads, 7*(4), 24–27.

Barr, S., & Stephenson, V. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads, 2*(1), 111–122.

Berthoz, A. (2009). *La simplexité*. Paris: Odile Jacob.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association (AERA'12)*, (pp. 1–25).

Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends, 62*(6), 563–573.

Cohen, L., Manion, L., & Morrison, K. (2007). *Research Methods in Education*. 6th Eds. London-New York: Routledge.

CSTA-ISTE (2011). *Operational Definition of Computational Thinking for K-12 Education*. Retrieved from id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf.

Dagienė, V., & Stupurienė, G. (2016). Bebras – A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education, 15*(1), 25–44.

Dufva, T., & Dufva, M. (2016). Metaphors of code: structuring and broadening the discussion on teaching kids to code. *Thinking Skills and Creativity*, 22, 97–110.

EUN (2015). *Computer programming and coding. Priorities, school curricula and initiatives across Europe*, European Schoolnet. Retrieved from www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf.

European Commission (2016). *A new skills agenda for Europe. Working together to strengthen human capital, employability and competitiveness*. Retrieved from ec.europa.eu/transparency/regdoc/rep/1/2016/EN/1-2016-381-EN-F1-1.PDF.

Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research, 87*(4), 834–860.

Gallese, V., & Lakoff, G. (2005). The Brain's Concepts: The Role of the Sensory-Motor System in Reason and Language. *Cognitive Neuropsychology,* 22, 455–479.

Grover, S., & Pea, R.D. (2012). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, XX, 1–6.

Grover, S. (2015). 'Systems of assessments' for deeper learning of computational thinking in K-12. In *Proceedings of the 2015 Annual Meeting of the American Educational Research Association (AERA '15)*, (pp. 1–9).

Hava, K., & Cakir, H. (2017). A systematic review of literature on students as educational computer game designers. In *Proceedings of EdMedia 2017*, (pp. 407–419). Washington, DC, USA.

Ioannou, A., & Makridou, E. (2018). Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work. *Education and Information Technologies, 23*(6), 2531–2544.

Korkmaz, Ö., Çakır, R., & Özden, M.Y. (2015). Computational thinking levels scale (CTLS) adaptation for secondary school level. *Gazi Journal of Educational Science, 1*(2), 143–162.

Korkmaz, Ö., Çakır, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). *Computers in Human Behavior, 72*, 558–569.

Korkmaz, Ö., & Bai, X. (2019). Adapting Computational Thinking Scale (CTS) for Chinese High School Students and Their Thinking Scale Skills Level. *Participatory Educational Research (PER)*, Vol. 6(1), 10–26.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads, 2*(1), 32–37.

[3]The references do not include the 53 papers that are the aim of our review, which are listed in Annex F2.

Lockwood, J., & Mooney, A. (2018). Computational Thinking in Secondary Education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools, 2*(1), 1–20.

MIUR (2015). *Piano Nazionale Scuola Digitale*. Retrieved from www.istruzione.it/scuola_digitale.

Olimpo, G. (2017). Dal mestiere dell'informatico al pensiero computazionale. *Italian Journal of Educational Technology*, *25*(2), 15–26.

Papert, S. (1980). *Mindstorms. Children, computers and powerful ideas*. New York: Basic Books.

Rivoltella, P.C. (2012). *Neurodidattica. Insegnare al cervello che apprende.* Milano: Raffaello Cortina.

Rivoltella, P.C., & Rossi, P.G. (2019). *Il corpo e la macchina. Tecnologia, cultura, educazione*. Brescia: Morcelliana.

Román-González, M. (2015). Computational Thinking Test: Design Guidelines and Content Validation. In *Proceedings of the 7th Annual International Conference on Education and New Learning Technologies (EDULEARN 2015)* (pp. 2436–2444). IATED, Barcelona, Spain.

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72 (July 2017), 678–691.

Román-González, M., Moreno-León, J., & Robles, G. (2017). Complementary Tools for Computational Thinking Assessment. In *Proceedings of International Conference on Computational Thinking Education (CTE 2017)*, S.C. Kong, J.Sheldon, & K.Y. Li (Eds.). *The Education University of Hong Kong* (pp. 154–159). Retrieved from www.eduhk.hk/cte2017/doc/CTE2017Proceedings.pdf.

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definitions. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. Canterbury: ACM.

Shute, V. J., Sun, Ch., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Sibilio, M. (2012). *Il corpo e il movimento nella ricerca didattica. Indirizzi scientificodisciplinari e chiavi teorico-argomentative.* Napoli: Liguori.

Trinchero, R. (2019). Problem solving e pensiero computazionale. Costruire sinergie tra concettualizzazione e codifica a partire dalla scuola primaria. *Form@re*, 19 (1), 78–90.

Varela, F.J. (1990). Il corpo come macchina ontologica. In M. Ceruti, & L. Preta, *Che cos'è la conoscenza.* Bari: Laterza.

Verborgh, R. (2013). Programming is an art [blog post]. Retrieved from ruben.verborgh.org/blog/2013/02/21/programming-is-an-art/

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715–728.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3)*,* 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366,* 3717–3725.

Annexes

Annex A: Data Collection
Annex B: Educational Contexts of CT Implementation
Annex C: Learning Contexts of CT Incorporation
Annex D: Contexts of Assessment
Annex E: Outcomes of the Study
Annex F: Reviewed Papers

Annex A: Data collection

Databases searched

This study reviews papers being published in journals of science, international-conferences proceedings, workshops and symposiums within the life-time span starting from the beginning of January 2006 and its end being December 2018.

To achieve this, ten big electronic databases related to 'educational contexts', 'digital technology' and 'social science' were used in the review in focus. More specifically, it made use of: SpringerLink, ACM (Association for Computing Machinery), Bio-Medical Library, ERIC (Education Resources Information Centre), IEEE Xplore Digital Library, Taylor & Francis Online, Wiley, LearnTechLib (Learning & Technology Library), Ingenta Connect and Science Direct.

Search term

The study was carried out using the key-words '*computational thinking*'. The study is restricted in the time period of January 2006 up to December 2018. This resulted in 3,547 papers/items being identified.

Papers selection to be included in the review[4]

The papers selected in the review were carefully chosen under specific criteria. Through a full screening of the identified papers and a close look at each paper's title, abstract and content, as well as an exclusion of those papers that were not relevant to the inclusion criteria, 53 papers were piled up. The inclusion criteria so as for the papers to be included in the review are reported below. The papers had to:

(a) explicitly refer the term *'computational thinking'* in title and/or abstract and/or keywords. Thus, papers that did not refer explicitly the term 'computational thinking' in title and/or abstract and/or keywords were excluded.

(b) be written in English language. Therefore, papers that were written in other languages apart from English (e.g., Ricci & Colombi, 2018) were removed.

(c) have a typical form of a scientific paper. Consequently, the papers in the below forms were excluded from the study:
- ✔ Abstracts (only) (e.g., Duncan, 2018; Amato & Acholonu),
- ✔ Invited keynotes/talks (e.g., Jordan, 2016), Keynote Addresses (e.g., O'Donnell, 2017; Astrachan, 2009) and keynote presentations (e.g., Corn, 2010),
- ✔ Demonstrations (e.g., Niu et al., 2015),
- ✔ Plenary sessions (e.g., Stein, 2006),
- ✔ Session presentations (e.g., Astrachan et al., 2014),
- ✔ Roundtables (e.g., Caristi et al., 2011),
- ✔ Posters (e.g., Russo et al., 2018; Meerbaum-Salant et al., 2015),
- ✔ Workshop descriptions (e.g., Settle, 2011; Easterbrook et al., 2010),
- ✔ Panel discussions (e.g., Maiorana et al., 2019; Kelleher et al., 2012),
- ✔ PhD dissertation proposals and doctoral consortiums (e.g., Labusch, 2018),
- ✔ Short papers and position papers (e.g., Hauswirth et al., 2017; Portelance & Bers, 2015),
- ✔ Work in progress papers (e.g., Psycharis & Kotzampasaki, 2017; Jenkins, 2015),
- ✔ Editorials (e.g., Edge, 2014),
- ✔ Talk brief descriptions (e.g., Brunvand, 2013),
- ✔ Books (e.g., Rich & Hodges, 2017).

(d) focus on CT and primary level education students (K-6). Consequently, papers in which the basic idea was not the cultivation of primary students' CT ability (e.g. Yildiz-Durak & Saritepeci, 2018) and/or did not have a didactic/teaching intervention for students' CT-development followed by a CT-assessment (e.g., Korucu et al., 2017; Marshall, 2011) were also removed from the study.

(e) present empirical data – emerged from methodologically sound empirical studies – either from Kindergarten students or from elementary students attending the grades 1–6 (K-6, less than 12 years old, according to the Greek educational system). In fact, studies:
- ✔ providing no adequate information about the research methodology, the participants and the research procedure used (e.g., Ball et al., 2012), or had just some informal/preliminary observations –without data – after application of a teaching intervention (e.g., Djurdjevic-Pahl, 2017) and/or presenting an overview of a project with general observations and results without empirical data (Serafini, 2011) were excluded from this review.
- ✔ reporting: pilot studies (e.g., Pugnali et al., 2017; Hoover et al., 2016), pilot experiences (e.g., Cabarello-González & Muñoz-Repiso, 2018), pilot implementations (e.g., Goodgame et al., 2018), preliminary studies (e.g., Moreno-León & Robles, 2015; Wilkerson-Jerde, 2014), preliminary investigations (e.g., Solitro et al., 2017), pilot tests and/with preliminary results/analysis (e.g., Jenson & Droumeva, 2016; Webb & Rosson,

[4]In this Annex, we refer to examples of excluded papers. For the full reference of those papers, see Annex F1.

2013), initial studies (e.g., Basu et al., 2016), initial implementations and evaluations (e.g., Choi et al., 2016) or early findings (e.g., Rode et al., 2015) were also excluded.

- ✔ referring to the following issues were removed:
  - teachers' education (e.g., Kaila et al., 2018), teacher's perceptions or practices on CT (e.g., Sands et al., 2018; Sadik et al., 2017), and/or teacher's conceptions and experiences on CT (e.g., Yadav et al., 2017),
  - mathematical algorithmic thinking (e.g., Benton et al., 2018) and/or algebraic thinking (Agatolio et al., 2018),
  - CT and handicap (e.g., Snodgrass et al., 2016),
  - CT and special education needs (e.g., Lechelt et al., 2018) or any kind of students disabilities or disorders, such as visual disabilities (e.g., Morrison et al., 2018) or Autism Spectrum Disorders (e.g., Munoz et al., 2018), and
  - literature reviews related to CT (e.g., Rich et al., 2017; Sullivan & Hefferman, 2016).

There were also six papers reported in two or more databases such as: Witherspoon et al., 2018 (Wiley and Ingenta Connect), Basu et al., 2017 (ACM, Ingenta Connect and Springer), Sung et al., 2017 (Ingenta Connect, ERIC and Springer), Liu et al., 2017 (Taylor & Francis and ERIC), Farris & Sengupta, 2016 (Wiley and ERIC), and Mouza et al., 2016 (Taylor & Francis and ERIC). Consequently, only one version of these papers were included in this review.

Moreover, there were two identical papers that were published two times in the same database in different date, which were kept as one, such as:

- ✔ Choi, Lee and Lee (2016) - Choi, Lee and Lee (2017) (both in Springer database),
- ✔ Pinto-Liorente et al. (2017) - Pinto-Liorente et al. (2018) (both in Springer database). Thus, only the earliest version of these papers were included in this review.

Furthermore, there were two papers related to the same research, which were published in different databases and were kept as one paper, such as:

- ✔ Karampa & Paraskeva (2018) (LearnTechLib and Springer). Here, the Springer's version was included in the review.

The total sum of papers that were collected from each database as well as the total sum of the ones that eventually became part of the review are shown in Table 1 of the main paper.

The whole sum of the papers under review are noted in the section F2 and can be found in Annex F. For their inclusion in this section, each of the papers was given a specific code ([1], [2],…[53]). Their citation is made within the paper and the Annexes through this code. Finally, with the above mentioned criteria in mind, out of 3,547 papers in total, 53 papers managed to meet the criteria under consideration and were the ones that were relevant in order for them to be fully textually reviewed.

To be more specific, twenty-eight papers were found in '*scientific journals*' ([2], [5], [6], [9], [11], [13], [14], [15], [17], [18], [19], [21], [26], [28], [31], [32], [33], [34], [36], [37], [40], [44], [46], [47], [48], [49], [51], [53]), sixteen were part of '*international conferences'* ([1], [3], [4], [8], [11], [12], [20], [24], [25], [27], [30], [35], [39], [43], [45], [52]), seven of them made their appearance in '*international symposiums*' ([1], [3], [4], [8], [11], [12], [20], [24], [25], [27], [30], [35], [39], [43], [45], [52]), and two of them participated in '*international workshops*' ([7], [22]).

Those twenty-eight papers being part of journals can be found in twenty-two different journals. The main themes of these journals revolved around computer, education and technology. Various databases such as Scopus, ProQuest, EBSCO, INSPEC, ERIC, Google Scholar, PsycINFO, Gale, ERA, SCImago and ERIH PLUS constitute the space where the journals are indexed/abstracted.

Annex B: Educational contexts of CT implementation

The below mentioned results emerged from the coding criteria set according the use of a data extraction proforma developed by considering the research question.

Table A1. Educational context of CT implementation

Educational Level	Location of the Study	Learning Subject	STEM Field	Duration of the Study
Primary: 31	School: 32	Computer science: 33	Yes: 44	Until 1 week: 18
Primary–Middle: 14	Summer camp: 7	Interdisciplinary: 6	No: 9	More than 1 week to 2 weeks: 5
Primary–Middle-High: 4	Workshop: 3	Robotics: 6		More than 2 weeks to 1 month: 2
Kindergarten–Primary: 3	Lab: 2	Edutainment: 3		More than 1 month to 6 months: 17
Kindergarten: 1	Home & School: 1	Science: 2		More than 6 months to 1 year: 3
	School in combination with lab and workshop: 1	Robotics and computer science: 1		More than 1 year: 4
	On-line: 1	Robotics and science: 1		Not mentioned: 4
	On-line in combination with school: 2	Mathematics: 1		
	Programming camp: 1			
	Not mentioned: 3			

Table A1 depicts an overview of the educational context in which CT has been implemented in primary education. Specifically, it presents: (a) the educational level, (b) the location of the studies taken place, (c) the learning subjects focusing the reviewed papers, (d) the number of studies related to STEM disciplines and education, and (e) the total duration of the studies, namely the duration between the beginning and the end of the studies.

Educational level

As it is shown in Table A1 (column 1), the educational level of the conducted studies do not concern only the primary school students (31 papers), but also students attending Kindergarten (1 paper), or both Kindergarten-Primary (3 papers) and Primary-Middle (14 papers) even Primary-Middle-High level of education (4 papers).

Location of the study

As for the location of these studies, most of them were conducted in the school settings (32 papers), such as the school's classroom or the school's computer lab. However, there are studies conducted in out-of-school settings such as summer camps (7 papers), workshops (3 papers), labs (2 papers), programming camps (1 paper), as well as online environments (1 paper). Furthermore, there were interventions taking place in more than just one setting, such as home and school (1 paper), online environments and school (2 papers), even at school in combination with lab and workshop (1 paper). Also, there are 3 papers that do not mention the place where the studies were conducted.

Learning subject

The studies focused on various disciplines. As presented in Table A1, a considerable number of papers (33 out of 53) refer to teaching interventions related to the subject of CS. In addition, there are studies focusing on the subject of Robotics (6 papers), Science (2 papers), Mathematics (1 paper), as well as combinations of subjects, such as Robotics and CS (1 paper), Robotics and Science (1 paper), even on Edutainment (3 papers) and interdisciplinary settings/approaches beyond STEM field (6 papers).

STEM field

From the aforementioned analysis, it is clear that the majority of studies (44 papers) focused on STEM disciplines for developing CT-competences to primary school students. However, there are some attempts (9 papers) to embed CT-concepts through: (a) playing digital games[35,46] or serious digital games with embodiment activities,[4] (b) interdisciplinary/combinatory approaches using mind maps, storyboards and visual environments in multidisciplinary activities,[17] as well as puzzles[10,11] and/or block-based programming languages such as Scratch for integrating CT-concepts in Science and Art Education,[36,37] and (c) teachers' training for implementing CT in various subjects.[29]

Duration of the study

Table A1 also shows that most studies lasted for up to one week (18 papers), while about equal number of papers (17 papers) show that the duration of the studies varies between one to six months. Moreover, there are diverse studies that lasted: more than one week to two weeks (5 papers), more than 2 weeks to 1 month (2 papers), more than 6 months to 1 year (3 papers), and more than 1 year (4 papers). Furthermore, there are four (4) papers that do not refer to the duration of the studies.

Annex C: Learning context of CT incorporation

Table A2 illustrates a summary of the learning context in which CT has been incorporated in primary education. Specifically, it depicts: (a) the main CT-concepts approached, (b) the way of integration, (c) the digital environments used throughout the teaching intervention, (d) the type of activities, (e) the use of a language for programming, and (f) the type of programming languages (in case it was used).

Table A2. Learning context of CT incorporation

Main CT-concepts Approached	Way of Integration	Digital Environment Used	Type of Activities
Abstraction: 41	Plugged programming activities: 15	Scratch: 16	Plugged: 34
Algorithms and procedures: 41	Game programming and game activities: 10	Alice: 4	Unplugged: 10
Control structures: 33	Robotics: 7	Kodu: 3	Combination: 8
Testing and verification: 19	No CS education activities: 6	AgentSheets: 3	Not mentioned: 1
Problem decomposition: 18	Unplugged programming activities: 4	ScratchJr: 2	
Parallelization: 16	Plugged and unplugged programming activities: 4	Lego WeDo software: 2	
Data analysis: 13	Paper activities: 4	ROBOTC Graphical: 2	
Simulation: 8	Simulation activities: 3	CTSiM: 2	
Data representation and analysis: 8		AgentCubes: 1	Use of Programming Language
Data collection: 6		BOTS: 1	Yes: 38
Automation: 2		CHERP: 1	No: 14
Not mentioned: 1		FormulaT Racing (FTR): 1	Not mentioned: 1
		Kodetu: 1	
		Lego Mindstroms EV3 software: 1	Type of Programming Language
		MiniColon game: 1	Visual: 36
		NAO platform: 1	Text based: 1
		Scratch 4 Arduino (S4A): 1	Combination: 1
		CompThink App: 1	Not mentioned: 1
		Lighbot: 1	No programming language: 14
		Story-Writing-Coding engine: 1	
		T-Maze: 1	
		ViMAP: 1	
		Zoombinis game: 1	
		Not mentioned: 1	

Main CT-concepts approached

As demonstrated in Table A2, the concepts of *abstraction* (41 papers), *algorithms and procedures* (41 papers) and *control structure* (33 papers) constitute the most common CT-concepts and competences approached in primary education. It is significant to be noted that abstraction includes/features the concepts/skills of patterns' recognition, patterns' abstraction and generalization, and variables as well (Wing, 2011; Czerkawski & Lyman, 2015:57; Csizmadia et al., 2015:7; Seiter, 2015:541). Moreover, the concepts of *testing and verification* (19 papers), *problem decomposition* (18 papers), *parallelization* (16 papers) and *data analysis* (13 papers) were approached to a lesser extent followed by the concepts of *simulation* (8 papers), *data representation and analysis* (8 papers), *data collection* (6 papers) and *automation* (2 papers). It should be also mentioned that the categorization of the aforementioned concepts/capabilities was based on the framework of CSTA and ISTE as stated to the work of Barr and Stephenson (2011:52; 2014:117).

Way of integration

Table A2 also illustrates the diverse ways through which CT has been integrated in the teaching practices for students' CT-skills development. These various ways are analysed across the following categories:

- *Plugged Programming Activities*: These activities used diverse digital environments including: (a) Scratch for fostering, developing, and assessing of students' CT-abilities via programming in STEM disciplines,[19,21,22,28] for evaluating the integration of CT in art education making use of technological resources, sensor cards and minicomputers,[37] and for synchronizing sprites through challenges on increasing complexity,[38] (b) Kodu for teaching lawfulness to students, getting them into the habit of reasoning about programs and predicting the behaviour of short Kodu programs,[41,42] (c) Scratch and CompThink App for teaching basic programming concepts to children with the parallel use of metaphors,[31] (d) Story-Writing-Coding engine aiming to compare the processes used by children to code an animated story with that of more experienced college students, (e) ScratchJr and Lightbot for exploring young learners' attitudes towards programming and considering the impact of programming approaches on developing of CT,[34] (f) Alice for employing a 3D framework of CT in order to learn how to storytelling via programming,[53] (g) Scratch and Alice for determining the effects of these programming teaching practices and tools on students' 'engagement, reflective thinking, problem-solving skills and CT comparatively',[51] (h) Kodetu for understanding how young programmers employ CT so as to tackle with a set of challenges,[14] and (i) BOTS for analysing students' problem solving behaviours in a programming game via debugging exercises.[26]
- *Game Programming and Game Activities:* Here, digital environments have been implemented in various ways for students' CT-cultivation through: (a) game play for teaching children an introduction to logical thinking and programming through MiniColon game,[4] for identifying specific CT-skills like problem decomposition, recognition of patterns, algorithmic thinking, as well as abstraction via Zoombinis game,[35] and for developing CT-strategies and CT-skills and practices via the constructionist video-game FormulaT Racing (FTR),[46] (b) game programming for assessing students' CT-skills in relation to three elements: 'computational notions', 'metacognition practices', and 'learning behaviours' via Scratch and Alice,[2] for investigating the degree of how effective pair programming is in K-12 through Alice,[13] for examining which programming notions learners make use of in order to create a game and in what ways these notions are related through Scratch 4 Arduino (S4A) and Scratch,[30] as well as for investigating whether there is a connection between more generic skills development and programming skills and CT – such as the so called twenty-first century skills of creative and critical thinking as well as problem solving – through Scratch and Kodu,[49] and (c) game and simulation design for utilizing collaboration as a means for allowing students to not only learn, but master and retain CT-patterns through AgentSheets and AgentCubes online environments.[23, 24, 50]
- *Robotics:* These activities used diverse digital programming environments supporting robotic education for students' CT-skills development, such as: (a) Lego WeDo software for addressing the issue of how educational robotics can be used to teach CT to young learners with the use of the LEGO WeDo kit,[3,32] (b) Lego Mindstorms EV3 software for teaching CT, programming, handling complexity and divide task into sub-tasks, project management, team work and robotics, (c) ROBOTC Graphical for examining how effective robotics programming learning contexts can be in developing wider CT competences[47] and/or determining if this participation is connected to any major positive or negative changes in motivational features such as interest, identity and competency claims,[48] (d) CHERP for engaging kindergarten pupils in learning CT, programming, robotics and problem-solving, as well as for exploring learning outcomes,[6] and (e) NAO platform in order that students make connections between robotics and everyday settings.[9]
- *No CS Education Activities:* In this context, students worked on activities not related to CS Education in order to enhance their CT-abilities. Specifically, researchers provided learners with opportunities to: (a) transform a material (for example, a computational programming language - ViMAP) into an expressive medium, meaning that the learner should be able to create a personally meaningful artefact (programming as an aesthetic experience,[15] (b) work on multidisciplinary activities[29] using mind maps, storyboards and visual programming languages – such as Scratch – to prove that CT can be implemented, not only in CS, but also in other fields,[17] (c) engage students in interdisciplinary programming activities – using Scratch – regarding science and arts,[36] (d) use tangible kits (e.g., CyberPLAYce) for storytelling and scenario/story creation,[39] and (e) perform tasks that Generation Z-kids typically do using their mobile gadgets in an alternative manner with the use of the natural environment beyond the schoolyard, making aware of CS-notions embedded in smartphone apps and promoting thus the CT acquisition.[45]
- *Unplugged Programming Activities:* Diverse activities for learners' CT-development without the use of computer were carried out in these studies. Specifically, researchers engaged students in various unplugged activities examining: (a) how embodied activities during an unplugged debugging intervention on a floor maze influences lower grade elementary school students' CT, problem-solving skills and self-efficacy,[1] (b) how students solve a problem with the use of programming concept cards, after their participation in a CT-problem activity that students are asked to solve with the use of coloured game tokens,[12] (c) Tangible technology Toys' potential (e.g., LittleBit) as a learning tool for CT for making circuits,[25] and (d) the thinking processes of young children working in pairs

when trying to program with the markers on the paper to follow 'Ozobot' (a small robot) in order to solve a given task focused on programming.[43]

- *Plugged & Unplugged Programming Activities:* There were also studies that used both plugged and unplugged activities for fostering CT in primary education. These activities included: (a) scaffolded programming exercises, and discussion of how each of them related to CS and other topics,[16] (b) tasks that are focused on programming through collaboration in a game known to the computer player in which learners first engaged in designing a high-level version of their algorithm during an 'unplugged pen and sheet' phase, and afterwards, they encoded their solution as a program that has the ability to be executed within a 'visual programming environment' (Scratch),[18] (c) many different degrees of 'embodied tasks' (i.e., full vs. low) to boost student's competence in programming and mathematics with their design aiming at integrating the computational aspect in non-programming disciplinary domains as a way of 'equipping' learners with skills needed, such as problem-solving skills, that can be 'transferred' to programming exercises,[40] and (d) tangible programming activities via playing of multilevel maze-escape games and then the creation of students' mazes, by manipulating a collection of wooden blocks.[44]
- *Paper Activities:* In these activities, the traditional method of integrating CT-elements to primary school students was used with the intention to: (a) analyse the extent to which the unplugged approach to the teaching of CT is effective enough,[7] trigger students' engagement in learning the skills so as to design an algorithm through a 'puzzle-based' algorithm learning program, investigating the effect of this program on learners' CT abilities,[10,11] and (c) discuss how a few changes in some CT tasks proposed during the Bebras challenge affect the solvers' performance.[27]
- *Simulation Activities:* There were also papers that engaged students in simulation activities aiming to: (a) cultivate a learner modelling approach to advocate scaffolding in CT that is adaptive, utilizing the 'Simulation and Modelling' (CTSiM), an open-ended learning environment, that boosts combined science learning and concepts of CT in classrooms of science,[5,52] and (b) encourage gifted students perform Scratch programming, making activities to collect and analyse data centring on the one to one (1:1) interrelations between batters and pitchers in order to simulate the process of professional baseball games and predict the results.[20]

Type of activities

In addition to the above analysis, Table A2 summarizes that most interventions were based on plugged activities (34 papers) for students' CT-development. Unplugged activities were also conducted to a lesser extent (10 papers). Moreover, there were efforts (8 papers) to combine plugged and unplugged activities for the same purpose.

Digital environments used

As shown in Table A2 (column 3), Scratch is the most popular digital environment to be used for primary school students CT-cultivation (16 papers) followed by Alice (4 papers), Kodu (3 papers), Agentsheets (3 papers), ScratchJr (2 papers), Lego WeDo software (2 papers), ROBOTC graphical (2 papers) and CTSiM (2 papers). There are more digital environments such as BOTS, CHERP and Lighbot that have been used once.

Use of programming language

As depicted in Table A2, there were 38 studies that used a programming digital environment for students' CT-nurture. Thus, children were required to work on a programming language for completing various activities.

Type of programming language

In addition to the aforementioned analysis, Table A2 presents that block-based visual programming languages constituted the majority of the programming environments used (36 out of 38). There was also one study that used a text-based programming language via the Story-Writing-Coding engine,[33] as well as a different one that uses two 'contexts of transfer', both 'text-based' and 'drag-drop coding environments'.[9]

Annex D: Contexts of assessment

The context of assessment of the aforementioned empirical studies is analysed according to: (a) the sample that is used – 'educational level and size', (b) the 'location of the study', and (c) the 'tools' that were used for the data to be collected. Table A3 illustrates the outcomes of all the reviewed studies.

Table A3. Context of the CT-assessment of the empirical studies

Sample: Participants	Sample Size: Number of Participants	Location of the Study	Tools for Data Collection
Primary: 31	Up to 10: 4	School: 32	Pre/post assessments: 28
Primary–Middle: 14	11–30: 7	Summer camp: 7	Artefacts' analysis: 17
Primary–Middle-High: 4	31–50: 9	Workshop: 3	Questionnaires: 17
Kindergarten–Primary: 3	51–100: 15	Lab: 2	Interviews/discussions: 15
Kindergarten: 1	More than 100: 16	Home and school: 1	Observations: 10
	Not mentioned: 2	School in combination with lab and workshop: 1	Log-files: 8
		On-line: 1	Pre/post surveys: 7
		On-line in combination with school: 2	Rubrics: 6
		Programming camp: 1	Various recordings (video, audio, screen-recordings, photographs): 5
		Not mentioned: 3	

As shown in Table A3, taking into consideration the 'educational level' and the 'size' of the '*sample*' enlisted in the reviewed papers, most students attending Primary School as well as Primary–Middle School were part of both 'small' and 'medium-scale' studies. As for the '*location*' of the interventions conducted, it seems that the diverse 'educational settings' applied were the expected ones, with the majority of them conducted into the school environment.

As for the '*research method*' followed, Table A3 presents that 'descriptive research methods' were utilized in most of the reviewed papers (30 papers) in order for researchers to investigate the diverse ways of CT integration in primary education settings. It is significant to be noted that there were also enough studies aiming not only to state hypotheses but test them as well through empirical studies using experimental research methods. Moreover, various *tools for data collection* were used, with pre or post assessments (28 papers), artefacts' analysis (17 papers), questionnaires (17 papers), and interviews/discussions (15 papers) being among the most common ones. Besides, a combination of different kind of such tools was used in the majority of the studies so that the results be more valid and reliable.

Annex E: Outcomes of the study

The analysis of the main outcomes emerging from the reviewed studies revealed the thematic topics, which are depicted in Table A4. As presented in Table A4, the main outcomes of the papers related to CT formed around 13 topics. Based on these topics, the most frequent finding relates to the cultivation of CT-skills ($f = 42$) followed by students' attitudes towards CT and CS ($f = 31$), as well as the position of CT in curricula ($f = 24$) and other skills cultivated beyond CT ($f = 20$). However, it was observed that the findings obtained had 'beneficial results' in the CT-context in general.

Table A4. Main findings emerged from the reviewed studies for CT implementation in primary education

Main Findings on CT for Primary Education	*f*
1. Findings on CT-skills:	**42**
Programming activities with visual programming environments improves CT	13
Game design/programming improves CT	7
Computer programming activities with robots improve CT	4
Activities with tangible kits improve CT	3
Simulation activities improve CT	3
Virtual robotics' programming improve CT	2
Embodied instructional activities improve CT	2
Puzzle solving improves CT	2
Metaphors improve CT	1
Coding an animated story improves CT	1
Constructionist video games improve CT	1
Non programming interdisciplinary activities improve CT	1
Experience-oriented outdoor activities on CS improve CT	1
Paper programming activities improve CT	1
2. Findings on students' attitudes:	**31**
Engagement/motivation/ enthusiasm	18
Interest	5
Fun/enjoyment	5
Toward computing	2
Self-expression	1
3. Findings on CT-position in curricula:	**24**
Appropriate ways for fostering CT in the classroom	12
Important elements for creation of CT learning tasks and curriculum	8
Necessity of CT's integration in primary education	4
4. Findings on other skills cultivated:	**20**
Programming skills	7
Problem-solving skills	4
Creative-thinking skills	4
Other 21st century skills	5
5. Findings on behavioural issues:	**15**
Collaboration/cooperation	9
Communication	4
Attention	1
Perseverance	1
6. Findings on gender issues:	**8**
Differences between sexes	5
No differences between sexes	3
7. Findings on CT-practices cultivated	**7**
8. Findings on factors affecting learning outcomes	**7**

9. Findings on evaluation of CT	**6**
10. Findings on domain knowledge beyond CT and CS	**5**
11. Findings on differences with traditional educational methods	**3**
12. Findings on differences between students' age	**2**
13. Findings on awareness of computing	**2**

The outcomes of the reviewed studies summarized in Table A4 are briefly discussed below according to the 13 different findings/themes obtained after the content analysis of the reviewed studies. The aforementioned themes relates to:

1. Findings on CT-skills cultivated:

- ✔ *Programming activities with visual programming environments improves CT:* Programming activities with 'visual programming languages' such as Scratch, Kodu or Alice revealed that pupils achieve substantial learning gains in CT-skills.[16,19,28,36,38,41,51] Also, approaches combining unplugged pen and paper and then visual programming activities,[18] graphical environments with the parallel use of diverse digital devices,[37] and/or visual programming activities designed into a blended learning environment allows students to familiarize with CT-concepts and skills.[22] Finally, these type of activities are also advantageous when taking into account agile software engineering methods,[17] students' transformative and fundamental nature of aesthetic experiences[15] and the application of design-based learning strategies into the teaching practice.[21]
- ✔ *Game design/programming improves CT:* Game programming activities develop students' CT-skills,[2,23,24, 50] while the most common concepts used on children's games is the 'sequence/event handling and conditionals', followed by 'threads' and 'operators'.[30] In addition, while students are engaged in game programming activities, they show an understanding of the association among CT, programming and 21st century skills.[49] Finally, in these type of activities, pair programming plays a central role for children's CT-skills cultivation and programming awareness building notably among less-experienced pupils.[13]
- ✔ *Computer programming activities with robots improve CT:* Computer programming activities with robots have demonstrated the contingent of NAO,[9] Lego Mindstorms EV3,[8] and Lego WeDo materials and software,[3, 32] to promote CT, since children have the opportunities to be engaged with various computational concepts – common in programming language – such as 'programming, coding, sequence, loops, events, parallelism, conditionals' and so on.
- ✔ *Activities with tangible kits improve CT:* Tangible technology toys/kits are effective learning tools, since these help students to acquire CT-concepts, practices and perspectives by: (a) snapping the diverse modules to make circuits,[25] (b) manipulating the tangible blocks to form their own programs,[44] as well as (c) creating stories/scenarios and storytelling by giving form to their thoughts through spatial construction.[39]
- ✔ *Simulation activities improve CT:* Students engaging in simulation activities enhance their CT-ability,[20,52] specifically if they receive a scaffolding for building more precise models, using modelling approaches efficiently and adopting more beneficial modelling manners.[5]
- ✔ *Virtual robotics' programming improve CT:* It has also been supported that students' engagement in scaffolded programming curricula, within the context of virtual robotics, is related to a growth in their 'generalizable' CT-awareness and skills.[47,48]
- ✔ *Embodied instructional activities improve CT:* Embodied instructional activities during unplugged debugging interventions influences lower grade elementary school students' CT, problem-solving skills and self-efficacy.[1] Similarly, the combination of 'full-embody activities' with the 'practice of computational perspective-taking' in solving problem in the area of mathematics improves CT-skills, the understanding of mathematics as well as programming skills.[40]
- ✔ *Puzzle solving improves CT:* 'Puzzle based algorithm' learning programs (PBA) are effective for cultivating students' CT, since during puzzle solving process, students solve puzzles using 'data collection, data representation, data analysis, abstraction, problem decomposition and algorithms' as CT problem solving elements.[10,11]
- ✔ *Metaphors improve CT:* There is a paper proposing that using a methodology based on metaphors and Scratch can meaningfully nurture pupils' CT-competences, but also that pupils have the capability to acquire basic concepts of programming.[31]
- ✔ *Coding an animated story improves CT:* It is also discussed that the coding of an animated story using a specific software such as the Story-Writing-Coding engine, which is based on a java language, improves students' CT-skills using fundamental CT concepts such as 'abstraction, decomposition, logical thinking and patterns'.[33]
- ✔ *Constructionist video games improve CT:* Constructionist video games improve students' CT-skills – such as 'testing and debugging computational constructions, iteratively developing and revising solutions and

identifying patterns' – since they are allowed to express their conceived gameplay strategies, thoughts, characters and so on, with the use of 'tailored' tools and illustrations.[46]

- ✔ *Non programming interdisciplinary activities improve CT:* Interdisciplinary activities without the use of programming develop students' CT-skills, since they focus on sharpening and applying CT-concepts in other domain activities.[29]
- ✔ *Experience-oriented outdoor activities on CS improve CT:* Experience-oriented outdoor activities on CS, such as 'Smartwalk', cultivate students' CT-competences, since they are strongly linked to the 'real life' environment. Students walk all over the place, get tired, observe cautiously and co-operate with their classmates with the intention to realize the CS-concepts adopted in smartphone applications and to encourage the CT-acquisition.[45]
- ✔ *Paper programming activities improve CT:* Unplugged approach using paper programming activities develop students' CT-skills and concepts such as 'problem decomposition, pattern recognition, abstraction, and algorithmic design'.[7]

2. *Findings on students' attitudes:*

- ✔ *Engagement/Motivation/Enthusiasm:* There are various studies referring to the view that CT-activities and approaches provoked students' engagement, motivation, enthusiasm and commitment. Specifically, these findings were observed in activities that students: (a) learn how to program helping them focus on problems,[31,36,51] (b) design/program a game with diverse software,[4,18,23,24] (c) construct and program their robots with specific visual programming interfaces,[6,8,32] (d) work with tangible kits,[25,44] (e) create stories,[45] (f) solve puzzles,[10] (g) express themselves with multiple modalities,[15] (h) work with technologies, programming and devices,[37] (i) work in blended learning environments,[22] as well as (g) cooperate with their classmates in order to realize the CS-concepts adopted in smartphone devices applications.[45]
- ✔ *Interest:* There are also studies discussing that: (a) design-based learning (DBL) can provoke students' self-interest and self-efficacy as it emphasizes the procedure of developing real materials,[21] (b) tasks performed with coding and devices provoke students' interest,[37] as well as (c) game activities,[18] (d) puzzle solving[10] and simulation-making activities increase their interest.[20]
- ✔ *Fun/Enjoyment:* Other studies allude to students' fun and enjoyment in activities related to: (a) debugging in gamified environments,[26] (b) visual programming,[36] (c) computer programming and robotics,[8] (d) storytelling,[39] and (e) programming with mazes using tangible electronic kits.[44]
- ✔ *Towards computing*: Other findings revealed that programming activities with visual programming languages have positive changes in students' attitudes toward computing.[28] Also, more experienced students than their classmate-partner managed better 'computer confidence' and greater 'positive attitudes' toward computing.[13]
- ✔ *Self-expression*: Finally, there was a study referring to the point that students express themselves imaginatively by creating stories and storytelling using tangible material with cards and icons.[39]

3. *Findings on CT-position in curricula:*

- ✔ *Appropriate ways for fostering CT in the classroom:* There are several studies proposing various ways for fostering CT in primary education via: (a) visual programming activities,[17,36,37] (b) game design/programming activities[23] and constructionist video games,[46] (c) robotics,[3,47,48] (d) story creation and storytelling using tangible kits,[39] (e) coherent classroom tasks that have the potential to integrate computational perspectives in an effective way into the current programming and mathematics curricula that occur in an interdisciplinary way,[4] (f) puzzle based algorithm learning tasks,[11] as well as (g) simulation activities.[20]
- ✔ *Important Elements for Creation of CT learning tasks and Curriculum:* There are important elements to be taken into account while creating CT learning tasks and curriculum. First of all, the forward tasks were not inferior to the reverse tasks in storytelling activities by programming.[53] As for the puzzles, it is needed to provide puzzles in order of difficulty from having a few to various constraints, increasingly difficult stages.[10] Another study notes that few changes in some CT tasks proposed during the Bebras challenge (e.g., examples and figures used) affects the solvers' performance.[27] Regarding the design of programming curricula in primary education, it is proposed for them to provide freedom and flexibility in the programming requirements,[46] with more time needed for students to develop and fully explore the complexity of the material in order to comprehend it in its totality,[6] as well as desirable learning experiences that foster computing participation.[28] Also, it is pivotal that the challenge set be adapted to programming learning platforms so as for assistance to be provided and positive encouragement, quickly detect spots where there are deviations from the norm and overcome obstacles such as the time-limit and the control-structures.[14] As for the robotics curricula, it is proposed to be strengthened and also be revised and refined the evaluation's instruments to better facilitate students' progress in CT, specifically in the subsequent areas: 'syntax, creativity, data processing, algorithmic thinking, inter-relations between the robotics programming environment and everyday reasoning'.[9]
- ✔ *Necessity of CT's integration in primary education:* There are important findings that that authors recommend the necessity of introducing CT at a very young age of students[24,45] and in central content areas, since it is of paramount importance in our era and modern society.[32] Programming should be included in the curriculum's

main goals in elementary education, as it broadens cognitive horizons, opening up to new dimensions and it has benefits that are not only limited to programming and computer skills but also to other areas of learner's capabilities across a variety of subject-specific matters.[49]

4. Findings on other skills cultivated:

- ✔ *Programming skills:* Apart from CT-skills cultivation, there were also studies that reported students' gains on programming skills while students were engaged in programming activities,[13,24,40,49] coding an animated story tasks,[33] and robotic activities.[8,48]
- ✔ *Problem solving skills:* Moreover, problem solving skills were visible whilst children engaged in game programming,[2,46] robotic[47] and simulation activities.[20]
- ✔ *Creative thinking skills:* Furthermore, the results showed an increase in students' creative thinking skills via visual programming activities, which provided freedom and flexibility,[49] game programming tasks,[2,24] as well as through art education making use of programming, technological resources, sensor cards and minicomputers, which allows students to play and create music.[37]
- ✔ *Other 21st century skills:* The findings also showed improvement on other 21st century skills. For example, students who worked with graphical programming environments were able to 'transfer' their knowledge from 'visual programming languages' to 'text-based' programming contexts.[19] Besides, they developed their digital competences,[37] and their reflective thinking skills as well.[51] They also showed an improvement in critical thinking, predictive thinking and evaluation, analytical thinking,[49] and metacognitive skills – such as planning, monitoring, and evaluation – after the intervention of designing and creating their own games.[2]

5. Findings on behavioural issues:

- ✔ *Collaboration/Cooperation:* The act of improving CT ability using design-based learning strategies,[21] blended learning environments,[22] game design activities,[2,18,50] robotics,[8] and simulation-making activities[20,24] enhance students' collaborative skills. It is also important to be noted that children's first computer experience and attitude toward cooperation influenced their partners/classmates in game programming activities with graphical environments.[13]
- ✔ *Communication:* Furthermore, the latent ability of students to learn through communication is one that should be exploited often and in as many classroom scenarios as possible. Reviewed studies showed that designed-based algorithm strategies,[21] visual programming[37] and game design activities[2,50] were effective methods for developing students' capacity to share, play, communicate, discuss their opinions and share their feedback with each other.
- ✔ *Attention:* It was also observable that all children paid attention during the lessons of programming activities with visual-programming environments regardless of their grade.[31]
- ✔ *Perseverance:* Finally, there was a study discussing that learning behaviours such as persevering is visible whilst children were coding their games.[2]

6. Findings on gender issues:

- ✔ *Differences between sexes:* Findings did indicate that boys were more confident towards their computing skills than girls in programming activities.[28] It is also visible in interdisciplinary activities without the use of programming, where boys achieved somewhat higher scores at both pre- and post-test time moments as well as greater benefits between the time moments.[29] Moreover, in a paper programming activity with Ozobot, lower grade boy pair-groups achieved greater scores on the sub-skills of CT than girl or a mixed pair-group of the same grade, while in the upper grades, this difference vanished.[43] In another study, comparisons between the genders demonstrated differences in the way that boys and girls used the programming constructs (e.g., the use of variables) both in Scratch and Alice graphical environments.[2] In contrast, another study unveiled that girls who took part in virtual robotics curriculum revealed relatively greater competence in the post-test than boys.[48]
- ✔ *No differences between sexes:* On the other hand, there were findings from which it can be deduced that there were no statistically important differences in stances toward computing[28] and/or in performance/efficacy between boys and girls in programming activities.[16] In addition, another study disclosed that the scores had no important difference between the boys and the girls of the school in total of six tasks storytelling activities by programming.[53]

7. Findings on CT-practices cultivated:

Students gained significant computational practices related to '*code organization'* and '*documentation'* in visual programming activities with Scratch.[28,36] Also, the context of robotics provided possibilities to students to get to know a set of computational practices such as '*testing* and *debugging*, *abstracting*, *reusing* and *remixed*, and *modularizing'*.[8,32] Likewise, players' 'programs' became more complex as they played a constructionist video-game, and their use of progressive CT-practices became more explicit, pinpointing to the fact of how the video game medium can help learners in cultivating and improving CT-practices.[46] Furthermore, CT-practices were improved through a synergistic learning via simulation activities,[52] as well as with the use of tangible technology toys/kits.[25]

8. Findings on factors affecting learning outcomes:

Regarding the findings of the reviewed papers, there were obtained significant factors that influence learning outcomes. First of all, prior STEM and computing experiences,[16,19] as well as the participation in previous CT-activities[12] were found to be strongly predicting the CT-learning results. Also, problem solving behaviours in debugging activities were highly correlated with students' own explanation ability, number of code edits, and prior experience in programming.[26] A finding that created more surprise was that more children with a higher ability in programming – as measured by non-verbal reasoning skill or game performance – engaged in a more intense manipulation of their programs.[34] Additionally, in another study, it was discussed that game design is a challenging and motivating factor for CT learning.[18] As for pair programming activities, it seemed that when pupils had a more positive stance to collaborative practices (relative to their pair), their programming knowledge dropped.[13]

9. Findings on evaluation of CT:

As for the CT-evaluation, reasoning problems appeared to be a promising tool for assessing CT in young programmers.[42] Furthermore, semi-open and open-tasks could evaluate further dimensional aspects of CT than the closed-tasks in activities by programming.[53] A proposal for multiple evaluation approaches is made in order to widely show the whole learning frame that the CT-process entails.[2] Moreover, game and simulation design is a way that has a beneficial impact on the educator, that is, the educator can fully understand and verify the learning of students' CT-skills.[50] In this context, a process also delineated for the creation of 'automated assessments' of indirect CT from gameplay behaviours. This work stands as a model for implicit 'GBLA' (game-based learning assessments) that can be used to disclose competences and knowledge through activity rather than depending on what learners are able to exert on typical evaluation models.[35] Finally, systematic research is suggested and proposed as a significant requirement, so that a CT-assessment instrument can be developed and a better valid and reliable tool can be provided towards the measurement of CT competency.[11]

10. Findings on domain knowledge beyond CT and CS:

Teaching computing in other domains beyond CT and CS seems to be a good indicator for gaining knowledge regarding other subjects by the students. Findings showed that children: (a) recognized that the necessity to work in an English interface involved the beneficial feedback so as to develop their English language skills and understanding;[46] (b) achieved all the educational goals of the topics across the activities occurring in a multidisciplinary frame;[17] (c) improved their mathematic understanding in full-embody activities;[40] (d) constructed more precise models, used modelling strategies in satisfactory ways, achieved a deeper comprehension of important science-concepts, and managed to convert their modelling skills and transfer them better to new case-scenarios;[5] as well as (e) presented learning gains in science-domain knowledge (e.g., understanding of diffusion through systems thinking) via simulation activities.[52]

11. Findings on differences with traditional educational methods:

There were also findings referring on differences with traditional educational methods. Specifically, a study revealed that 'Design-Based Learning' (DBL) is more valuable than direct teaching approaches in order to broaden students' CT ability.[21] Another study noted that the learning gain achieved from the group using the Kinect game – an interactive serious game – was notably greater than the group that was introduced to the traditional educational method.[4] Finally, puzzle based algorithm learning had better influence on improving CT competency in comparison to the conventional algorithm method of learning.[11]

12. Findings on differences between students' age:

There were also findings on differences between students' age. Firstly, in a study aiming to detect young learner's ways of perceiving and approaching programming in two tools with contrary programming interfaces and consider the effect of programming approaches on cultivating CT, it was observed that students belonging to age six and seven years had a different way of interaction with the ScratchJr-like programming interface in contrast with the Lightbot interface.[34] Moreover, in a programming task using *Ozobot,* results indicated that the youngest duos (grade 2–4) approached the programming task in a non-structured way. In case they succeeded, the goal was reached by trial and error. As for grade 4, pair groups were able to recognize that the case was divided into subtasks. The couples could set goals for these tasks, tested a targeted sequence of actions and connected pieces of information. In addition, the couples in the highest grades (5–6) were able to formulate how to connect different pieces of information.[43]

13. Findings on awareness of computing:

As found in a study, CT education that has 'DBL' as its main basis (designed-based learning) appears to assist in altering learners from passive to active learners as 'prosumers'. Actually, DBL encourages the learners to have a more active awareness of the usefulness of computer tools that can help them in creative ways.[21] Furthermore, 'outdoor activities' on CS, like *'Smartwalk',* seems to get students to perceive digital devices in an alternative way, devices that they use during their everyday life. The main idea is to cultivate in children the feeling and the ability to have an outlook on the world that looks at things from a different angle and through a different lens – the viewpoint of a computer scientist. Instead of being a producer (taking the role of an 'artist' or 'engineer'), it is about experimenting, reflect upon and discuss concepts that are used in digital technology (undertaking the role of a 'philosopher').[45]

Annex F: Reviewed Papers

Excluded Papers

Here we present the reference of the excluded papers mentioned in Annex A.

Agatolio, Fr., Albanese, F., & Moro, M. (2018). How an Ambitious Informatics Curriculum Can Influence Algebraic Thinking of Primary School Children. In *Proceedings of the International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP '18*, (pp. 354–365). St. Petersburg, Russia.

Amato, D. A., & Acholonu, U. (2017). Designing Blended Learning Models to Support Computational Learning: Minecraft Edition (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17*, (pp. 733–733). Seattle, Washington, USA.

Astrachan, O. (2009). A new way of thinking about computational thinking. *Journal of Computing Sciences in Colleges*, *25*(1), 6–6.

Astrachan, O. (2014). Diverse Learners, Diverse Courses, Diverse Projects: Learning from Challenges in New Directions. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE '14*, (pp. 177–178). Atlanta, Georgia, USA.

Ball, C., Moller, F., & Pau, R. (2012). The Mindstorm Effect: A Gender Analysis on the Influence of LEGO Mindstorms in Computer Science Education. In *Proceedings of the Workshop in Primary and Secondary Computing Education, WIPSCE '12*, (pp. 141–142). Hamburg, Germany.

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, *11*(13), 1–35.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. *Digital Experiences in Mathematics Education, 3*(2), 115–138.

Brunvand, Er. (2013). Arts/Tech Collaboration with Embedded Systems and Kinetic Art. In *Proceedings of the 40th International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '13*, (pp. 1–1). Anaheim, California, USA.

Cabarello-González, Y. A., & Muñoz-Repiso, A. G.-V. (2018). A robotics-based approach to foster programming skills and computational thinking: Pilot experience in the classroom of early childhood education. In *Proceedings of the 6th International Conference on Technological Ecosystems for Enhancing Multiculturality, TEEM '18*, (pp. 41–45). Salamanca, Spain.

Caristi, J., Sloan, J., Barr, V., & Stahlberg, E. (2011). Starting a Computational Science Program. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '11*, (pp. 3–4). Dallas, Texas, USA.

Choi, I., Kopcha, T., Mativo, J., Hill, R., Hodge, E., Shin, S., Way, B., McGregor, J., Kim, S., Choi, J., & Bae, Y. (2016). Learning Computer Programming in Context: Developing STEM-integrated Robotics Lesson Module for 5th Grade. In *Proceedings of Society for Information Technology & Teacher Education International Conference 2016,* (pp. 68–74). Savannah, GA, United States.

Corn, M. (2010). Keynote Presentation: AI, Biomedicine and the NIH. *In Proceedings of the 1st ACM International Health Informatics Symposium, IHI '10*, (pp. 1–1). Arlington, Virginia, USA.

Djurdjevic-Pahl, A., Pahl, C., Fronza, I., & El Ioini, N. (2017). A Pathway into Computational Thinking in Primary Schools. In: Wu TT., Gennari R., Huang YM., Xie H., Cao Y. (eds), *Emerging Technologies for Education*, SETE 2016, Lecture Notes in Computer Science, vol 10108. Springer, Cham.

Duncan, C. (2018). Reported Development of Computational Thinking, Through Computer Science and Programming, and its Benefits for Primary School Students: (Abstract Only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18*, (pp. 275–275). Baltimore, Maryland, USA.

Easterbrook, St., Mens, K., & Zschaler, St. (2010). Second International Workshop on Software Research and Climate Change. In *Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering, JCSE '10*, (pp. 449–450). Cape Town, South Africa.

Edge, K. (2014). Editorial. *Educational Assessment, Evaluation and Accountability, 26*, 317–318.

Goodgame, C. (2018). BeeBots and Tiny Tots. In E. Langran & J. Borup (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference* (pp. 1179–1183). Washington, D.C., United States: AACE.

Hauswirth, M., Adamoli, An., & Reza-Azadmanesh, M. (2017). The program is the system: introduction to programming without abstraction. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, (pp. 138–142). Koli, Finland.

Hoover, A. K., Puttick, G., Barnes, J., Tucker-Raymond, E., Fatehi, B., Harteveld, C., Moreno-Leon, J. (2016). Assessing Computational Thinking in Students' Game Designs. In *Proceedings of the 2016 Annual Symposium on Computer-*

Human Interaction in Play Companion Extended Abstracts, CHI PLAY Companion '16, (pp. 173–179). Austin, Texas, USA.

Jenkins, Cr. (2015). A Work in Progress Paper: Evaluating a Microworlds-based Learning Approach for Developing Literacy and Computational Thinking in Cross-curricular Contexts. In *Proceedings of the 10th Workshop in Primary and Secondary Computing Education, WiPSCE '15*, (pp. 61–64). London, United Kingdom.

Jenson, J., & Droumeva, M. (2016). Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study. *The Electronic Journal of E-Learning, 14*(2), 111–121.

Jordan, M. I. (2016). Invited Talk: On Computational Thinking, Inferential Thinking and Data Science. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '16*, (pp. 47). Pacific Grove, CA, USA.

Kelleher, T., Collins, S., Dierbach, Ch., Jerome, G., & Kleinsasser, W. (2012). Interdisciplinary experiments in and perspectives on computational thinking (panel discussion). *Journal of Computing Sciences in Colleges, 27*(3), 50–51.

Kaila, E., Laakso, M. J., & Kurvinen, E. (2018). Teaching future teachers to code — Programming and computational thinking for teacher students. In *Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics*, *MIPRO '18*, (pp. 677–682). Opatija, Croatia.

Korucu, A. T., Gencturk, A. T., & Gundogdu, M. M. (2017). Examination of the Computational Thinking Skills of Students. *Journal of Learning and Teaching in Digital Age, 2*(1), 11–19.

Labusch, Am. (2018). Fostering Computational Thinking through Problem-Solving at School. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18*, (pp. 276–277). Espoo, Finland.

Lechelt, Z., Rogers, Y., Yuill, N., & Nagl. L. (2018). Inclusive Computing in Special Needs Classrooms: Designing for All. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, CHI '18*, (pp. 1–12). Montréal, QC, Canada.

Maiorana, Fr., Richards, Gr., Lucarelli, Ch., Berry, M., & Ericson, B. (2019). Interdisciplinary Computer Science Pre-service Teacher Preparation. In the *Proceedings of the 25th annual conference on Innovation and Technology in Computer Science Education*, (pp. 332–333). Aberdeen, Scotland, UK.

Marshall, K. S. (2011). Was that CT? Assessing Computational Thinking Patterns through Video-Based Prompts. *Online Submission*. *Paper presented at the Annual Meeting of the American Educational Research Association*, (pp. 1–12). New Orleans, LA, USA.

Meerbaum-Salant,O., Haberman, B., & Pollack. P. (2015). "Computer Science, Academia and Industry" as pedagogical model to enhance Computational thinking. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15*, (pp. 341–341). New York, USA.

Moreno-León, J., & Robles, G. (2015). Computer programming as an educational tool in the English classroom : A preliminary study. In *Proceedings of Global Engineering Education Conference, EDUCON '15*, (pp. 961–966). Tallinn, Estonia.

Morrison, C., Villar, N., Thieme, A., Ashktorab, Z., Taysom, E., Salandin, O., Cletheroe, D., Saul, Gr., Blackwell, A.F., Edge, D., Grayson, M., & Zhang, H. (2018). Torino: A Tangible Programming Language Inclusive of Children with Visual Disabilities. *Human–Computer Interaction*, 1–49.

Munoz, R., Villarroel, R., Barcelos, T. S., Riquelme, F., Quezada, A., Bustos-Valenzuela, P. (2018). Developing Computational Thinking Skills in Adolescents With Autism Spectrum Disorder Through Digital Game Programming. *IEEE Access, 6*, 63880–63889.

Niu, Sh., Esakia, A., & McCrickard, D.S. (2015). Exploring Computer Science Topics with Programmable Smartwatvhes. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '15*, (pp. 440–440). Kansas City, MO, USA.

O'Donnell, C. (2017). Teaching game design and development: why bringing "the social" into the classroom is key. *Journal of Computing Sciences in Colleges, 33*(1), 5–6.

Portelance, D.J., & Bers, M.U. (2015). Code and Tell: Assessing Young Children's Learning of Computational Thinking Using Peer Video Interviews with ScratchJR. In *Proceedings of the 14th International Conference on Interaction Design and Children, IDC '15*, (pp. 271–274). Medford, MA, USA.

Psycharis, S. & Kotzampasaki, E. (2017). A didactic Scenario for Implementation of Computational Thinking using Inquiry Game Learning. In *Proceedings of the International Conference of Education and ELearning, ICEEL '17*, (pp. 26–29). Bangkok, Thailand.

Pugnali, A., Sullivan, A., & Bers, M.U. (2017). The impact of user interface on young children's computational thinking. *Journal of Information Technology Education: Innovations in Practice, 16*, 171–193.

Ricci, M., & Colombi A.E. (2018). Dalla mano al video. Esperienze e osservazioni di costruzione del pensiero astratto, analitico e computazionale nella formazione linguistica della scuola primaria. *Italiano LinguaDue, 9*(2), 291–314.

Rich, K., Strickland, C., & Franklin, D. (2017). A Literature Review through the Lens of Computer Science Learning Goals Theorized and Explored in Research. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17*, (pp. 495–500). Seattle, WA, USA.

Rich, P., & Hodges, Ch. (2017). *Emerging Research, Practice, and Policy on Computational Thinking*. Springer: Switzerland.

Rode, J.A., Weibert, A., Marshall, A., Aal, K., Rekowski, T.V., Mimoni, H., & Booker, J. (2015). From Computational Thinking to Computational Making. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '15,* (pp. 239–250). Osaka, Japan.

Russo, D., Missiroli, M., & Ciancarini, P. (2018). Poster: a Conceptual Model for Cooperative Thinking. In *Proceedings of the ACM/IEEE 40th International Conference on Software Engineering: Companion Proceeding*, (pp. 157–158). Gothenburg, Sweden.

Sadik O., Leftwich A.O., Nadiruzzaman H. (2017). Computational Thinking Conceptions and Misconceptions: Progression of Preservice Teacher Thinking During Computer Science Lesson Planning. In: Rich P., Hodges C. (eds), *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations*. Springer, Cham.

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. In: Khine, M. (eds), *Computational Thinking in the STEM Disciplines*. Springer, Cham.

Serafini, G. (2011). Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects*. In Proceedings of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP - 2011*, (pp. 143–154). Bratislava, Slovakia: Springer.

Settle, Am. (2011). The Computational Thinking across the Curriculum Workshop. In *Proceedings of the 12th Annual Conference on Information Technology Education, SIGITE '11*, (pp. 311-312). West Point, New York, USA.

Snodgrass, M.R., Israel, M., & Reese, G.C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers & Education, 100*, 1–17.

Solitro, U., Pasini, M., De Gradi, D., & Brondino, M. (2017). A Preliminary Investigation on Computational Abilities in Secondary School. In *Proceedings of the International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP '17*, (pp. 169–179). Helsinki, Finland.

Stein, L.An. (2006). A small footprint curriculum for computing (and why on earth anyone would want such a thing). *Journal of Computing Sciences in Colleges, 21,* 3–3.

Sullivan, F.R., & Heffernan, J. (2016). Robotic Construction Kits as Computational Manipulatives for Learning in the STEM Disciplines. *Journal of Research on Technology in Education*, 1–25.

Webb, H., & Rosson, M.B. (2013). Using Scaffolded Examples to Teach Computational Thinking Concepts. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13*, (pp. 95–100). Denver, Colorado, USA.

Wilkerson-Jerde, M.H. (2014). Construction, categorization, and consensus: Student generated computational artifacts as a context for disciplinary reflection. *Educational Technology Research and Development, 62*(1), 99–121.

Yadav, A., Gretter, S., Good, J., & McLean, T., (2017). Computational Thinking in Teacher Education, In: P.J. Rich, C.B. Hodges (eds.), *Emerging Research, Practice, and Policy on Computational Thinking, Educational Communications and Technology: Issues and Innovations*, Springer, Charms.

Yildiz-Durak, H., & Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers and Education, 116*, 191–202.

Included Papers

Finally, here we present the 53 papers we included in the systematic literature review, with the code number we used to identify them.

[1] Ahn, J. H., Mao, Y., Sung, W., & Black, J. B. (2017). Using Embodied Instructions in Children's Programming Education. In *Proceedings of the Society for Information Technology & Teacher Education International Conference, SITE '17*, (pp. 19–26). Austin, TX, United States: LearnTechLib.
[2] Allsop, Y. (2018). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer*, 1–26, in press.
[3] Angeli, Ch., & Makridou, E. (2018). Developing Third-Grade Students' Computational Thinking Skills with Educational Robotics. In *Proceedings of the Society for Information Technology & Teacher Education International Conference, SITE '18*, (pp. 1–8). Washington, D.C., United States: AACE.
[4] Ayman, R., Sharaf, N., Ahmed, Gh., & Abdennadher, Sl. (2018). MiniColon; Teaching Kids Computational Thinking Using an Interactive Serious Game. In *Proceedings of the Joint International Conference on Serious Games, JCSG '18*, (pp. 79–90). Darmstadt, Germany: Springer.
[5] Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a Computational Thinking-based science learning environment. *User Modeling and User-Adapted Interaction, 27*(1), 5–53: Springer.
[6] Bers, M.U., Flannery, L., Kazakoff, E.R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157.
[7] Brackmann, Ch.P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. In *Proceedings of the 12th Workshop in Primary and Secondary Computing Education, WiPSCE '17*, (pp. 65–72). Nijmegen, Netherlands: ACM.
[8] Chaudhary, V., Agarwal, V., Sureka, P., & Sureka, A. (2016). An Experience Report on Teaching Programming and Computational Thinking to Elementary Level Children using Lego Robotics Education Kit. In *Proceedings of the 8th International Conference on Technology for Education, T4E '16*, (pp. 38–41). Mumbai, India: IEEE.
[9] Chen, G., Shen, J., Barth-Cohen, L., Jiang, Sh., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education, 109*, 162–175.
[10] Choi, J. & Lee, Y. (2015). Reflection of Informatics Gifted Students to Puzzle Based Algorithm Learning for Improving Computational Thinking. In *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2015,* (pp. 69–74). Kona, Hawaii, United States: AACE.
[11] Choi, J., Lee, Y., & Lee, E. (2016). Puzzle Based Algorithm Learning for Cultivating Computational Thinking. *Wireless Personal Communications*, 1, 1–15.
[12] Conde, M. Á., Fernández-Llamas, C., Rodríguez-Sedano, F.J., Guerrero-Higueras, Á.M., Matellán-Olivera, V., & García-Peñalvo, F.J. (2017). Promoting Computational Thinking in K-12 students by applying unplugged methods and robotics. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality, TEEM '17*, (pp. 1–6). Cádiz, Spain: ACM.
[13] Denner, J., Werner, L., & Campe, S., Ortiz, E. (2014). Pair Programming: Under What Conditions Is It Advantageous for Middle School Students? *Journal of Research on Technology in Education, 46*(3), 277–296.
[14] Eguiluz, A., Guenaga, M., Garaizar, P., & Olivares-Rodriguez, Cr. (2018). Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. *IEEE Transactions on Emerging Topics in Computing*, 1–6, in press.
[15] Farris, A.V., & Sengupta, P. (2016). Democratizing Children's Computation: Learning Computational Science as Aesthetic Experience. *Educational Theory*, *66*(1–2), 279–296.
[16] Feldhausen, R., Weese, J.L., & Bean, N.H. (2018). Increasing Student Self-Efficacy in Computational Thinking via STEM Outreach Programs. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGSE '18*, (pp. 302–307). Baltimore, MD, USA: ACM.
[17] Fronza, I., Ioini, N.E., & Corral, L. (2017). Teaching Computational Thinking Using Agile Software Engineering Methods: A Framework for Middle Schools. *ACM Transactions on Computing Education, 17*(4), 1–28.
[18] Gardeli, A., & Vosinakis, S. (2017). Creating the computer player: an engaging and collaborative approach to introduce computational thinking by combining 'unplugged' activities with visual programming. *Italian Journal of Educational Technology, 25*(2), 36–50.
[19] Grover, Sh., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237.
[20] Hong, C., Jeon, Y., Kim, T., & Kim, J. (2014). Design and Application of Computational Thinking-Type Simulation Making Education Utilizing Scratch for Elementary Gifted Students. In *Proceedings of E-Learn:*

World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2014, (pp. 848–855). New Orleans, LA, USA: AACE.

[21] Jun, S., Han, S., & Kim, S. (2017). Effect of design-based learning on improving computational thinking. *Behaviour & Information Technology, 36*(1), 43–53.

[22] Karampa, V., & Paraskeva, F. (2018). A Motivational Design of a Flipped Classroom on Collaborative Programming and STEAM. In *Proceedings of the International Workshop on Learning Technology for Education in Cloud*, LTEC '18, (pp. 226–238). Žilina, Slovakia: Springer.

[23] Koh, K.H., Endo, Y., Repenning, A., Nickerson, H., & Motter, P. (2013). Will it Stick? Exploring the Sustainability of Computational Thinking Education Through Game Design. *In Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, *SIGCSE '13,* (pp. 597–602). Denver, Colorado, USA: ACM.

[24] Lamprou, A., Reppening, A., & Escherle, N.A. (2017). The Solothurn Project — Bringing Computer Science Education to Primary Schools in Switzerland. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '17*, (pp. 218–223). Bologna, Italy: ACM.

[25] Lin, V., & Shaer, O. (2016). Beyond the Lab : Using Technology Toys to Engage South African Youth in Computational Thinking. *In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, (pp. 655–661). San Jose, CA, USA: ACM.

[26] Liu, Z., Zhi, R., Hicks, A., & Barnes, T. (2017). Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education, 27*(1), 1–29.

[27] Lonati, V., Malchiodi, D., Monga, M., & Morpurgo, A. (2017). How presentation affects the difficulty of computational thinking tasks: an IRT analysis. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research, Koli Calling '17*, (pp. 60–69). Koli, Finland: ACM.

[28] Mouza, C., Marzocchi, A., Pan, Y.C., & Pollock, L. (2016). Development, Implementation, and Outcomes of an Equitable Computer Science After-School Program: Findings From Middle-School Students. *Journal of Research on Technology in Education, 48*(2), 84–104.

[29] Ouyang, Y., Hayden, K.L., & Remold, J. (2018). Introducing Computational Thinking through Non-Programming Science Activities. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGSE '18*, (pp. 308–313). Baltimore, MD, USA: ACM.

[30] Papavlasopoulou, S., Giannakos, M.N., & Jaccheri, L. (2018). Discovering children's competences in coding through the analysis of Scratch projects. In *Proceedings of the IEEE Global Engineering Education Conference, EDUCON '18*, (pp. 1127–1133). Santa Cruz de Tenerife, Canary Islands, Spain: IEEE.

[31] Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2018). Can computational thinking be improved by using a methodology based on metaphors and Scratch to teach computer programming to children? *Computers in Human Behavior*, 1–25, in press.

[32] Pinto-Llorente, A.M., Casillas-Martín, S., Cabezas-González, M., & García-Peñalvo, F.J. (2017). Building, coding and programming 3D models via a visual programming environment. *Quality & Quantity*, 1–14.

[33] Price, C.B., & Price-Mohr, R.M. (2018): An Evaluation of Primary School Children Coding Using a Text-Based Language (Java). *Computers in the Schools*, 1–19.

[34] Rose, S.P., Habgood, J.M.P., & Jay, T. (2017). An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking. *The Electronic Journal of e-Learning, 15*(4), 297–309.

[35] Rowe, E., Asbell-Clarke, J., Baker, R., Gasca, S., Bardar, E., & Scruggs, R. (2018). Labelling Implicit Computational Thinking in Pizza Pass Gameplay. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI EA '18*, (pp. 1–6). Montréal, QC, Canada: ACM.

[36] Sáez-López, J.M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129–141.

[37] Sáez-López, J.M., & Sevillano-García, M.L. (2017). Sensors, programming and devices in Art Education sessions. One case in the context of primary education/ Sensores, programación y dispositivos en sesiones de Educación Artística. Uncaso en el contexto de Educación Primaria. *Cultura y Educación, 29*(2), 350–384.

[38] Seiter, L. (2015). Using SOLO to Classify the Programming Responses of Primary Grade Students. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE '15*, (pp. 540–545). Kansas City, MO, USA: ACM.

[39] Soleimani, A., Green, K.E., Herro, D., & Walker, I.D. (2016). A Tangible, Story-Construction Process Employing Spatial, Computational-Thinking. In *Proceedings of the ACM SIGCHI Conference on Interaction Design and Children 2016*, (pp. 157–166). Manchester, United Kingdom: ACM.

[40] Sung, W., Ahn, J., & Black, J.B. (2017). Introducing Computational Thinking to Young Learners: Practicing Computational Perspectives Through Embodiment in Mathematics Education. *Technology, Knowledge and Learning, 22*(3), 443–463.

[41] Touretzky, D.S., Gardner-McCune, C., & Aggarwal, A. (2016). Teaching "Lawfulness" With Kodu. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education, SIGCSE '16*, (pp. 621–626). Memphis, TN, USA: ACM.

[42] Touretzky, D.S., Gardner-McCune, Ch., & Aggarwal, A. (2017). Semantic Reasoning in Young Programmers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGGSE '17*, (pp. 9–14). Seattle, WA, USA: ACM.

[43] Van der Linde-Koomen, D., Van der Aar, N., & Voogt, J. (2018). How Children Use Computational Thinking Skills When They Solve a Problem Using the Ozobot. In *Proceedings of EdMedia: World Conference on Educational Media and Technology*, (pp. 2151–2157). Amsterdam, Netherlands: AACE.

[44] Wang, D., Wang, T., & Liu, Z. (2014). A Tangible Programming Tool for Children to Cultivate Computational Thinking. *The Scientific World Journal*, 1–10.

[45] Weigend, M. (2017). Smartwalk: Computer Science on the Schoolyard. In *Proceedings of the IFIP World Conference on Computers in Education, WCCE '17*, (pp. 525–535). Dublin, Ireland: Springer.

[46] Weintrop, D., Holbert, N., Horn, M.S., & Wilensky, U. (2016). Computational Thinking in Constructionist Video Games. *International Journal of Game-Based Learning*, *6*(1), 1–17.

[47] Witherspoon, E.B., Higashi, R.M., Schunn, Ch.D., Baehr, E.C., & Shoop, R. (2017). Developing Computational Thinking Through a Virtual Robotics Programming Curriculum. *ACM Transactions on Computing Education, 18*(1), 1–20.

[48] Witherspoon, E.B., Schunn, Ch.D., Higashi, R.M., & Shoop, R. (2018). Attending to structural programming features predicts differences in learning and motivation. *Journal of Computer Assisted Learning, 34*, 115–128.

[49] Wong, G.K.-W., & Cheung, H.-Y. (2018): Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments*, 1–14.

[50] Worrell, B., Brand, C., & Repenning, A. (2015). Collaboration and Computational Thinking. A Classroom Structure. In the *2015 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC '15*, (pp. 183–187). Atlanta, Georgia, USA: IEEE.

[51] Yildiz-Durak, H. (2018). The Effects of Using Different Tools in Programming Teaching of Secondary School Students on Engagement, Computational Thinking and Reflective Thinking Skills for Problem Solving. *Technology, Knowledge and Learning, 1–17*.

[52] Zhang, N., & Biswas, G. (2018). Understanding Students' Problem-Solving Strategies in a Synergistic Learning-by-Modeling Environment. In *Proceedings of the International Conference on Artificial Intelligence in Education, AIED '18*, (pp. 405–410). London, United Kingdom: Springer.

[53] Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research*, *53*(4), 562–590.