Mapping Computational Thinking through Programming in K-12 Education: A Conceptual Model based on a Systematic Literature Review

Computational Thinking (CT) through programming attracts increased attention as it is considered an ideal medium for the development of 21st century skills. This intense attention leads to K-12 initiatives around the world and a rapid increase in relevant research studies. However, studies show challenges in CT research and educational practice. In addition, the domain has not been mapped to facilitate comprehensive understanding of the domain challenges and development of CT curricula. The purpose of this study is to develop a conceptual model based on a systematic literature review that maps the CT through programming in K-12 education domain. The proposed Computational Thinking through Programming in K-12 education (CTPK-12) conceptual model emerges from the synthesis of 101 studies and the identification of CT Areas. The proposed model consists of six CT Areas (namely Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building) and their relationships. The model could aid domain understanding and serve as a basis for future research studies. In addition, it could support the integration of CT into K-12 educational practices, providing evidence to educational stakeholders and researchers as well as bringing closer research, practice and policy.

Keywords: Elementary education; Secondary education; 21st century abilities

1. Introduction

Computational Thinking (CT) has its roots in 1980s with Papert's (1980) attempts to introduce programming to young students. Later in 2006, Wing (2006) defines CT as a process that "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science". CT is considered a necessary skill for everyone (Wing, 2006) and an ideal medium for the development of 21st century skills (Lye & Koh, 2014; Grover & Pea, 2013).

After about fifteen years of renewed interest in CT, the domain of CT research is beginning to mature. It is indicative that a large number of studies focusing on CT have been published in recent years (T.-C. Hsu, Chang, & Hung, 2018). This large body of literature indicates challenges in particular areas including (a) developing widely accepted assessment methods and frameworks that encompass the complexity of CT (Brennan & Resnick, 2012; Denner, Werner, & Ortiz, 2012; Denning, 2017; Fronza, Ioini, & Corral, 2017; Grover et al., 2017; Grover, Pea, & Cooper, 2015; Moreno León, Robles, & Román González, 2015; Zhong, Wang, Chen, & Li, 2016), (b) designing theoretically-based approaches that align learning strategies with CT (Dolgopolovas, Dagienė, Jasutė, & Jevsikova, 2019) and (c) identifying the knowledge needed to teach CT (Angeli et al., 2016; Cooper, Grover, Guzdial, & Simon, 2014) and methods by which support to teachers is provided (Yadav, Stephenson, & Hong, 2017). A comprehensive mapping of the domain would enable better understanding of challenges and guide future research.

We acknowledge there are several literature reviews that examine the whole domain from different perspectives as well as propose frameworks and definitions. Researchers in these studies review the literature in order to derive insights on CT through programming for K-12 curriculum (Lye & Koh, 2014), to understand the development and application of CT in education (Hsu et al. 2018), to facilitate the CT learning and assessment within K-12 curricula (Shute, Sun, & Asbell-Clarke, 2017) and to support educators in developing CT tasks and programs (Kalelioglu, Gulbahar, & Kukul, 2016). Despite all these efforts, a comprehensive mapping of the domain is still lacking.

In addition, efforts to integrate CT in schools are taking place worldwide responding to societal need for 21st century skills (Buitrago Flórez et al., 2017; Y.-C. Hsu, Irie, & Ching, 2019; Passey, 2017). At the same time, many undergoing initiatives promote CT by providing curriculum suggestions (Csizmadia et al., 2015), CT and programming tools and resources (García-Peñalvo & Mendes, 2018). However, educators do not have an overall map of the CT through programming in K-12 education domain to help them design CT curricula. This is evident from the fact that several studies highlight that teachers lack clear understanding of how CT could be effectively integrated into K-12 educational practices (e.g., Denning, 2017; Grover & Pea, 2013; Yadav et al., 2017).

One way to move the research domain forward and facilitate CT educational practices is to systematically study the existing literature and create a conceptual model that maps the domain. Conceptual model development is «the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication» (Mylopoulos, 1992). Conceptual models offer in developing domain understanding through

aiding reasoning about the domain, communicating the domain details and documenting the domain for future reference (Gemino & Wand, 2004). In addition, a conceptual model could be an effective roadmap between what we know and what we need to know, providing a firm foundation for advancing the domain knowledge (Webster & Watson, 2002). A conceptual model of CT though programming in K-12 education could provide such a foundation, helping researchers better understand the domain and its challenges through a holistic approach and identify areas that have already been covered by research and areas where more research is needed. In addition, a conceptual model serves as a point of agreement (Mylopoulos, 1992) and thus could support CT teaching and learning in K-12 education by providing a reference point for teachers.

The goal of this study is to develop a conceptual model of CT through programming in K-12 education based on a systematic literature review. This model could aid domain understanding and serve as a basis for future studies. It could also support researchers to focus on significant research gaps in their CT studies, having an up-to-date synthesis of the relevant literature. In addition, it could support the integration of CT into K-12 educational practices, providing evidence to teachers and policy-makers as well as bringing closer research, practice and policy.

Conceptual model development includes the identification of the concepts and relationships of the domain and their visual representation (Wand & Weber, 2002). In this respect, we systematically review the literature and record all topics of interest to researchers e.g. assessment, professional development to support teachers, relevant tools etc., as discussed in the scientific publications. These topics are then grouped into concepts that we call CT Areas. The proposed CTPK-12 conceptual model presents these CT Areas and their relationships.

2. Background

CT is the thought process that involves solving problems and designing model systems by utilizing Computer Science (CS) core concepts (Wing, 2008). CT draws on concepts from CS but is a fundamental skill for everyone. Wing (2006) argues that "to reading, writing and arithmetic, we should add computational thinking to every child's analytical ability". Aho (2012) defines CT as "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms." Many other definitions exist in the literature.

CT definitions can be classified into two main categories**:** generic definitions that focus on CT as a thought process (Román-González, Pérez-González, & Jiménez-Fernández, 2017) and definition models that describe what CT entails. The second category comprises efforts that develop models describing CT elements. For example, Barr & Stephenson's (2011) model presents CT concepts and capabilities in the context of various subjects. Angeli et al. (2016) develop a conceptual framework that describes CT skills. CSTA & ISTE (2011) provide an operational definition of CT describing CT characteristics and attitudes. Selby's (2013) definition model depicts CT thought processes. Weintrop, Beheshti, et al. (2016) develop a model that presents CT practices. Brennan & Resnick's (2012) CT framework describes CT concepts, practices and perspectives. Shute et al. (2017) develop a competency model that can be used in CT assessment. Kalelioglu, Gulbahar, & Kukul (2016) develop a framework that describes CT skills, considering CT to be a problem-solving process. Zhang & Nouri (2019) extend Brennan & Resnick's (2012) framework by including CT skills. A summary of CT elements described in some of the definition models is presented in (Table 1). We select to present the specific models as they are highly cited in the literature, cover an extensive period of time and are developed based on different approaches (e.g. systematic literature review, previous authors' studies, literature summary, meeting procedures).

Table 1

CT elements in CT definition models

Barr & Stephenson (2011)	Brennan & Resnick (2012)	Selby (2013)	Angeli et al. (2016)	Shute et al. (2017)
Abstraction Analysis and Model validation Simulation Data collection, analysis and representation	Abstracting and modularizing	Ability to think in abstractions	Abstraction	Abstraction • Data collection and analysis • Pattern recognition • Modeling
Problem decomposition		Ability to think in terms of decomposition	Decomposition	Decomposition

Algorithms and procedures Control structures Parallelization Automation	Computational concepts (mapping to Scratch programming blocks such as sequences, loops etc.)	Ability to think algorithmically	Algorithms • Sequencing • Flow of control	Algorithms • Algorithm design • Parallelism • Efficiency • Automation
Testing and verification	Testing and debugging	Ability to think in terms of evaluations	Debugging	Debugging
		Ability to think in terms of generalizations	Generalization	Generalization
	Being incremental and iterative			Iteration
	Reusing and remixing			
	Expressing			
	Connecting			
	Questioning			

CT is widely associated with programming (Voogt, Fisser, Good, Mishra, & Yadav, 2015), although programming is not the only approach examined in the literature. However, the relationship between CT and programming is not clear (Passey, 2017). On the one hand, programming offers a necessary mechanism for implementing CT concepts and practices (Basogain, Olabe, Olabe, & Rico, 2018; Brennan & Resnick, 2012) and is considered to be a key tool for supporting the cognitive tasks involved in CT (Grover & Pea, 2013). On the other hand, CT gives programming a new direction, making it a means of understanding the digital world while at the same time promoting teaching programming in K-12 educational settings (Kafai, 2016; Lye & Koh, 2014). This dual association between CT and programming can be described as follows: Programming supports the development of CT while CT provides to programming a new upgraded role.

Integration of CT into the curriculum is an important goal for many countries around the world (Heintz, Mannila, & Farnqvist, 2016). In the US, most CS curricula include CT concepts such as problem decomposition, debugging, abstraction, and algorithms (Grover et al., 2017). The Israeli curriculum exposes students to CT with the aim of introducing students to logical and algorithmic thinking (Bargury et al., 2012). There are also efforts underway in other countries, including Poland, New Zealand, Estonia, Finland, Sweden, Norway and South Korea (Heintz et al., 2016).

The strong interest in CT is also indicated by the increased number of literature reviews in recent years (Table 2). Some of these reviews focus on a specific topic of CT domain, such as assessment, while others cover multiple topics. Reviews that cover multiple topics can be classified in three categories: a) studies aiming to develop a definition model (e.g. Kalelioglu et al., 2016; Shute et al.; 2017) b) studies reviewing the literature to provide insights on teaching and learning CT (e.g. Grover & Pea 2013; Lye & Koh, 2014; Buitrago Flórez et al., 2017) and c) studies aiming to analyze CT research (e.g. Hsu et al., 2018). Despite all this work reviewing various aspects of CT through programming in K-12 education, a conceptual model of the domain is still missing.

Table 2

Literature Reviews in CT domain

Review	**Main Contribution**	**Scope**	**CT approach**	**Main focus on educational level**	**Studies included**
(Grover & Pea, 2013)	Review CT definitions, the rationale for integrating CT into	General	Programming	K-12	Undefined

	K-12 education, tools for CT development and assessment, and provide information on what CT entails and how is integrated in K-12 education.				
(Lye & Koh, 2014)	Review the trends of empirical research in the development of CT through programming in K-12 education such as programming environments, learning outcomes and approaches, and derive insights on K-12 curriculum.	General	Programming	K-12	Empirical higher education and K-12 articles
(Kalelioglu et al., 2016)	Review theoritical basis, definition, CT elements, population, type of research design, and develop a framework that includes notion, scope and elements of CT.	General	Programming and unplugged methods	K-12	Higher education and K-12 articles
(Buitrago Flórez et al., 2017)	Review challenges faced by early programmers, programming languages and pedagogical tools, and provide an overview of how programming is being taught in K-12 and higher education.	General	Programming	K-12 and higher	Journal articles, reviews, proceedings, short communications, and governmental standards
(Shute et al., 2017)	Review CT definitions and characteristics, interventions, assessments and models, and develop a CT competency model.	General	Programming and other approaches	K-12	Conceptual papers and empirical studies
(T.-C. Hsu et al., 2018)	Review learning strategies, teaching instruments, programming languages and course types, and analyze the evolution of CT research.	General	Programming and other approaches	All educational levels	SCI and SSCI journal articles
(Ching, Hsu, & Baldwin, 2018)	Review the technologies used for developing CT in young learners.	Focused on technologies	Programming and other approaches	K-12	Undefined
(Da Cruz Alves, Gresse Von Wangenheim, & Hauck, 2019)	Review the automatic assessment tools used to analyze artifacts in order to assess CT skills.	Focused on automatic assessment	Programming	K-12	K-12 and higher education articles
(Zhang & Nouri, 2019)	Review the CT skills that can be obtained through Scratch in K-9 education and extend Brennan	Focused on CT elements	Scratch programming	K-9	K-9 empirical studies

	& Resnick's (2012) framework.				

3. Study design

3.1 Goal and research questions

The study goal is the development of a conceptual model for CT through programming in K-12 education. The model aims to describe the CT Areas and the relationships between them. The conditions in which CT is integrated in K-12 education such as policies and issues regarding national curricula are falling out of scope of the model.

The research questions are:

RQ1. What are the areas of CT through programming in K-12 education domain?
RQ2. What are the sub-areas of each CT Area?
RQ3. How do CT Areas relate to each other?

3.2 Method

In order to develop a conceptual model for CT through programming in K-12 education we proceed to the following two steps proposed by (Wand & Weber, 2002): a) elicit the domain knowledge and b) visualize the domain knowledge. Fig. 1 presents the study method in terms of steps conducted and relevant results. We apply the Webster and Watson's (2002) systematic literature review approach for the elicitation of the domain knowledge (CT Areas and their relationships). This includes a structured approach to identifying sources and a concept-centric approach to presenting the results. We started by applying the PRISMA Statement (Moher, Liberati, Tetzlaff, & Altman, 2009) for the study selection phase. We then, proceed to the coding scheme identification phase, in which we identify the CT Areas that serve as a coding scheme for the data extraction phase. The data extraction phase aims to identify the sub-areas of each CT Area and the CT Areas' relationships**.** The process concludes with the visualization of the data extraction phase results. The whole process evolved into iterative phases where searches led to new selected studies that were being analyzed, leading to revised CT areas, sub-areas and relationships. The steps followed in this study are further elaborated below.

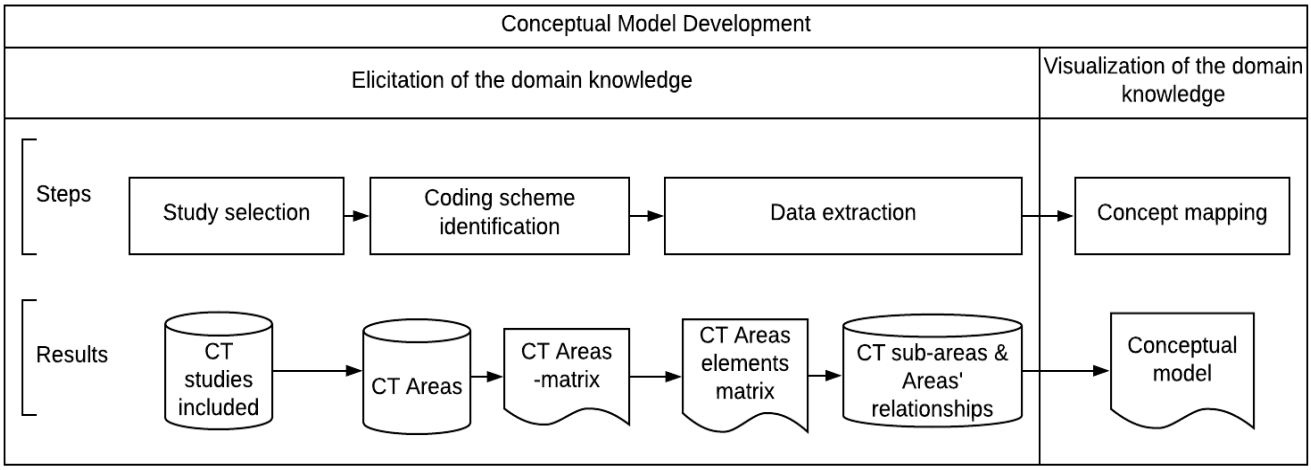


Fig. 1. Method

3.2.1 Elicitation of the domain knowledge

3.2.1.1 Study selection

We carry out the study selection presented in (Fig. 2), adapting the PRISMA Statement (Moher et al., 2009). Specifically, we adapt the PRISMA flow diagram (Fig. 2) by placing *additional records identified* in *included phase,* as we identified these studies by examining the selected studies as proposed by Webster & Watson (2002).
The selection of studies included in the review is a critical factor for the validity of the review. For this reason, the authors identified the search keywords and criteria together but worked individually to screen the studies and apply the inclusion and exclusion criteria. During this process a few conflicts emerged, which were solved through discussions until agreement was reached. The results of this phase are presented in detail in the supplementary material (all supplementary material is listed in Appendix B).

The sub-steps of study selection phase are outlined in the following sub-sections.

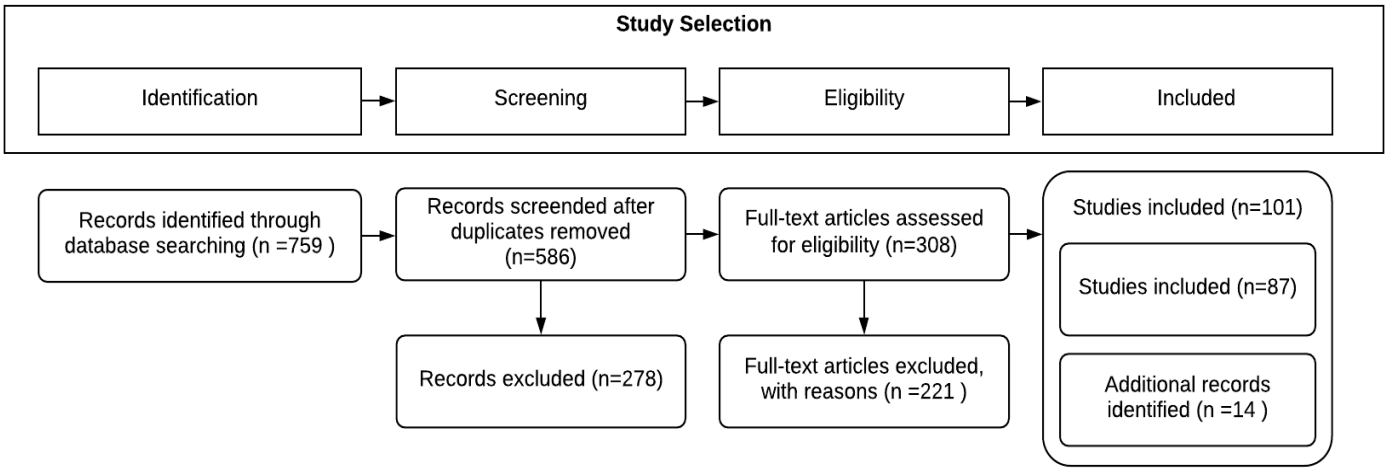


Fig. 2. Process applied for study selection adapted by (Moher et al., 2009).

3.2.1.1.1 Identification

The relevant studies were detected using keywords in the scientific databases Web of Science and Scopus. Specifically, we searched the phrase "computational thinking", quotations included, with a time constraint of 2006 onwards. The year 2006 was chosen as it was then that the term "Computational Thinking" was re-introduced by Wing (2006). In Scopus we included title, abstracts, keywords and in Web of Science we defined category as Education Educational Research. In both databases we included only articles and reviews. Searches took place from March 2018 to October 2019 maintaining the aforementioned structure. In total, three searches took place that resulted in 759 studies, 499 articles in Scopus database and 260 in Web of Science database.

3.2.1.1.2 Screening

In this sub-step we screened the studies retrieved from the previous step after we removed 173 duplicates. To this end, we read all the titles and abstracts and we removed the studies that were not written in English or were not fully available. We also excluded short papers. This sub-step resulted in 308 studies remaining.

3.2.1.1.3 Eligibility

During this sub-step we filtered out the studies retrieved from the screening process by examining the full-texts and applying the following inclusion and exclusion criteria.

The Inclusion Criteria suggest: a) studies should be published in journals; b) studies can be conceptual papers, opinion articles and empirical studies, as the incorporation of conceptual papers in addition to empirical studies broadens the scope of the review by including theoretical frameworks and future directions; c) the focus should be on CT in K-12 education and should involve programming; d) in the case of empirical CT studies, in addition participants should be K-12 students, K-12 pre-service teachers or K-12 inservice teachers.

The Exclusion Criteria suggest studies are excluded when a) they do not specifically focus on CT in K-12 education, such as studies that focus on higher education b) they do not specifically focus on CT through programming, such as studies where examination approaches focus on tangible artifacts, board games, exhibits etc., and c) they refer to CT only in their introduction or background and not in their results or they measure something other than CT.

3.2.1.1.4 Included

Subsequently, the studies were further processed by reviewing their citations (backward) and identifying articles that cite them (forward). The process resulted in the collection of 14 additional studies including 2 gray literature materials. Finally, 101 studies (Appendix A) were included in the study.

3.2.1.2 Coding scheme identification

To determine the areas of CT through programming in K-12 domain that serve as our coding scheme, we apply conventional content analysis. Conventional content analysis is suggested when existing theory is limited and does not involve a predefined coding scheme but one that derives from text analysis (Hsieh & Shannon, 2005). We choose conventional content analysis because of the lack of a conceptual model describing the domain. Initially, we read all

full-text articles in order to approach the domain as a whole. Then we carefully read each article and highlight keywords that imply a concept/area. Keywords are combined together, providing categories of the coding scheme. For example, keywords "assessing the development of Computational Thinking", "assessment" (Brennan & Resnick, 2012), "assess and evaluate", "assessment" (Zhong et al., 2016) are grouped and eventually led to adding "Assessment Area" in the coding scheme. Subsequently, we sort the studies in these categories. During this phase the coding scheme evolves by adding new categories or merging and splitting existing ones. The phase leads to the identification of the final categories, which from now on will be referred to as *CT Areas* and serve as the coding scheme and as the concepts of the conceptual model.
Consequently, we compile a concept-matrix or CT Area-matrix, which is a matrix listing the CT Areas where each article contributes. This matrix is available in the supplementary material (Appendix B). In this way we transit from an author-centric to a concept-centric approach, as suggested by Webster and Watson (2002) (Table 3).

Table 3
Approaches to Literature Reviews adopted from Webster and Watson (2002)

Concept-centric	**Author-centric**
Concept X [Author A, Author B]	Author A [Concept X, Concept Y]
Concept Y [Author A, Author C]	Author B [Concept X, Concept W]

3.2.1.3 Data extraction

During this phase, we sort the selected studies into the coding scheme. In this respect, we use a table for each CT Area available in the supplementary material (Appendix B). When we insert a study into the table, we also record the area's elements that appear in the study (Fig. 3). Subsequently, we compare every element with all other elements. The elements with clear match with other elements constitute a sub-area. For example, in Assessment Area, "project analysis" (Brennan & Resnick, 2012) and "examination of artifacts for CT patterns" (Denner et al., 2012) are included in the "Artifact analysis" sub-area. Sub-areas consisting of only one element and low-frequency (<2 studies) sub-areas, are only presented in the supplementary material (Appendix B) and not included in the model.

Subsequently, we use a table for each CT Area in order to record evidence in studies that suggest relationships between sub-areas (Fig. 4) and therefore Areas. We then group these evidences and conclude to the relationships between areas.

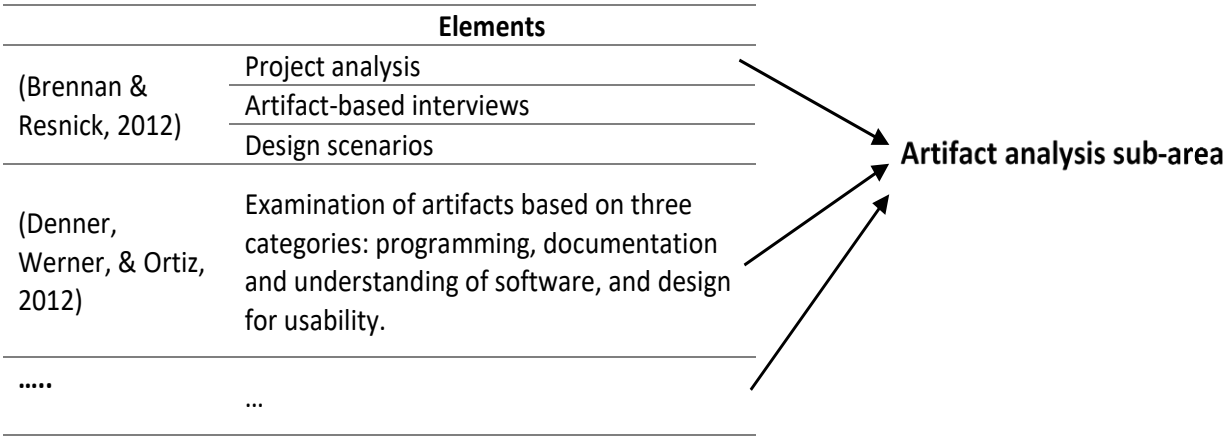
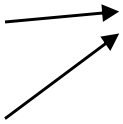


Fig. 3. Example of elements recording and sub-areas identification.

	Evidence of relationships
(Brennan & Resnick, 2012)	Project analysis (Assessment sub-area) of the CT concepts within Scratch projects (Knowledge base sub-area)
….	….



Relationship between Assessment Area & Knowledge Base Area

Fig. 4. Example of evidence recording and relationship identification.

3.2.2 Visualization of the domain knowledge

3.2.2.1 Concept mapping

In this step, we use concept mapping as proposed by Siau & Tan, (2005) for visualizing the concepts (CT Areas) and relationships of the domain, the identification of which is described in section 3.2.1. We create a visualization of the conceptual model depicting CT Areas as nodes. At each node, we note the sub-areas of each CT Area, identified in the previous phase. Finally, we depict the relationships between CT Areas as links. We then place a label to each link to explain the relationship.

3.3 Study limitations

We acknowledge that this study has a number of limitations. First, the proposed model is based on the analysis of studies written in English. Second, searches for studies were conducted in only two scientific databases, namely Web of Science and Scopus. Third, searches included only articles published in journals. Although, we eventually included some conference papers and gray literature identified through manual inspection of the references of the selected studies, still the majority of the selected literature includes journal articles. Fourth, searches were conducted with a time constraint of 2006 onwards. Thus, the model is based exclusively on the research conducted since 2006 and not on the initial stages of CT research. Fifth, non-inclusion of studies on the basis of quality criteria (Section 3.2.1.1.3) prevents the presentation of all conducted research. Finally, subjectivity combined with the small number of authors (only two) constitutes an additional limitation of the study. Although we applied a systematic method (presented in Section 3.2) we had to make subjective choices regarding e.g. grouping the elements, defining the relationships based on the recorded evidence, naming the CT Areas and sub-areas, and defining exclusion criteria for selecting sub-areas that are finally included in the model.

4. Results

4.1 Identification of CT Areas

The analysis of the 101 studies during the coding scheme identification phase resulted in the determination of six CT Areas described in (Table 4). The CT Area-matrix in which we note the areas that CT studies contribute is available in the supplementary material (Appendix B). CT studies attempt to address the challenges of CT through programming in K-12 education domain by focusing on these areas that repeatedly appear in the selected studies.

Table 4

CT Areas

Knowledge Base Area	CT measurable elements and their classification.
Assessment Area	Assessment methods and frameworks for measuring CT through programming in K-12 education.
Learning Strategies Area	Learning strategies leveraged to enhance students' CT learning through programming in K-12 education.

Factors Area	Factors related to CT through programming acquisition in K-12 education.
Tools Area	Tools that are used or specifically developed for teaching and learning CT through programming in K-12 education.
Capacity Building Area	Capacity building needed for teaching CT through programming in K-12 competently.

The percentage of studies by CT Areas to which they contribute is depicted in (Fig. 5). We categorize the studies into two groups 2006-2014 and 2015-2019. As shown in Fig. 5, Assessment and Tools are the two most popular areas that gather the greatest interest of researchers in both periods. Assessment Area is coming first across the two timelines (27.9% in period 2006-2014, 25.6% in period 2015-2019) followed by Tools Area (20.9% in both periods). During period 2006-2014 Knowledge Base Area is coming third (18.6%) while in period 2015-2019 the percentages of studies aimed at defining CT fall to 8.5% placing the area as the one with the least interest. On the contrary, the percentage of studies that focus on Learning Strategies increases from 9.3% during period 2006-2014 to 17.1% during period 2015-2019, placing Learning Strategies in the third place of researchers' interest in the selected studies. Respectively for the Capacity Building Area the percentage of studies that focus on this area increases from 9.3% during period 2006-2014 to 14.7% during period 2015-2019, placing Capacity Building in the fourth place of interest followed by Factors. These results indicate that as the field matures efforts still focus on assessment and tools but the focus shifts beyond the definition of CT on more tangible issues such as Learning Strategies, Capacity Building and Factors.

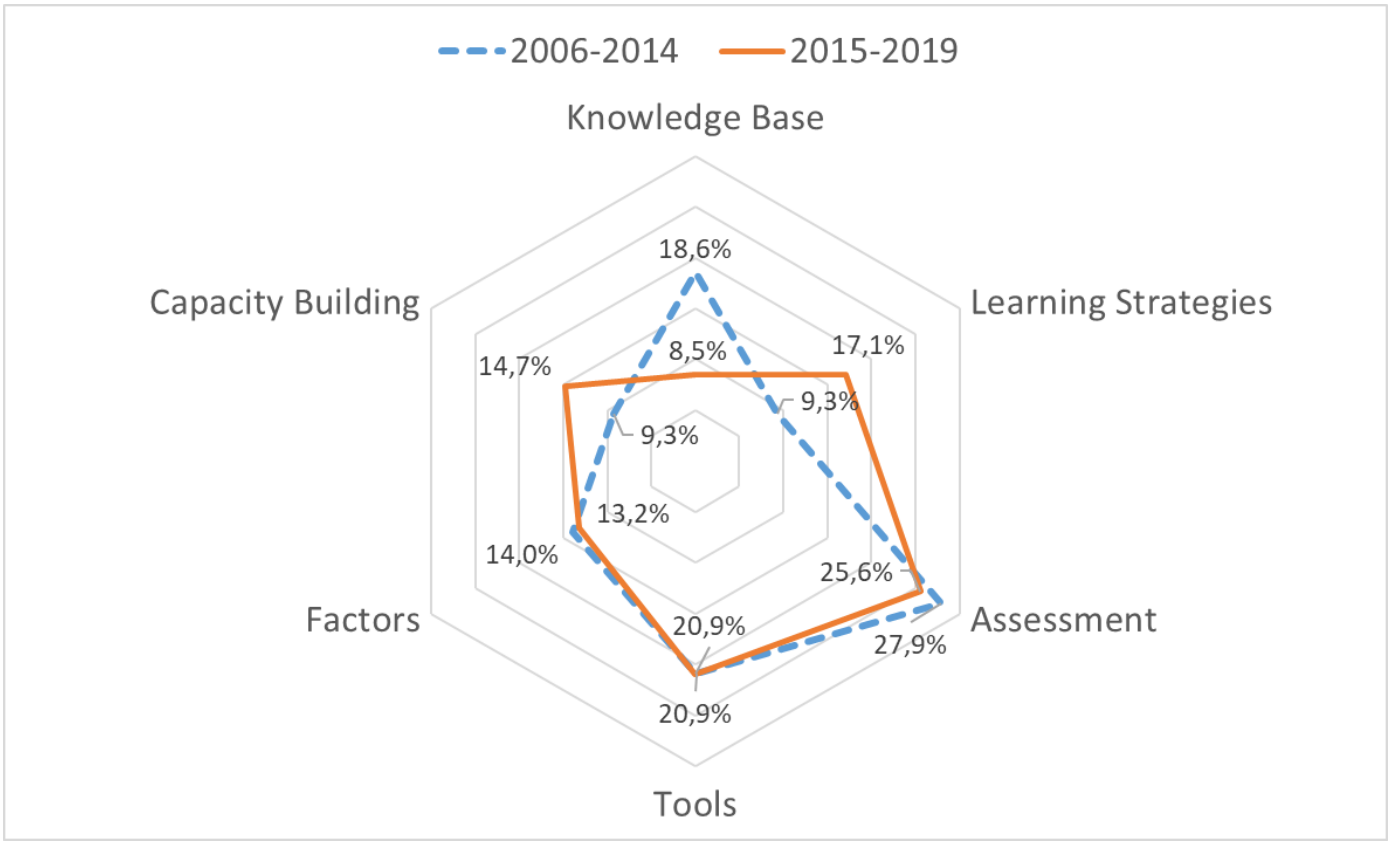


Fig. 5. Percentage of studies by CT Areas to which they contribute in the periods 2006-2014 and 2015-2019.

References to 2019 actually refer to period January 2019 to October 2019.

4.2 CT Areas

4.2.1 Knowledge Base Area

Knowledge Base Area is at the core of the domain. 57 of the 101 studies are included in this CT Area. Researchers in these studies either propose a framework or a definition to identify and classify measurable elements of CT, or

simply assess CT elements in order to assess CT. Based on CT frameworks we examined CT elements in the selected studies. We classify Knowledge Base Area in five sub-areas: concepts, skills, practices, perspectives and attitudes (Table 5). Fig. 6 presents the number of studies by CT element.

The results of the CT knowledge base analysis in the selected studies, include various **CT elements** and **terms describing classifications of CT elements** such as skills, capabilities, perspectives, attitudes, practices, characteristics, concepts, facets and thought processes. Some of these terms are often presented with different meaning.

In addition, several CT elements such as Abstraction, Algorithms, Decomposition, Data representation, Testing, Evaluation, Debugging, Generalization, Iteration appear to be classified in various ways including CT skills, CT concepts, CT practices or thought processes. For example, abstraction occurs as the thought process of "the ability to think in abstractions" (Selby, 2013), as "the skill to decide what information about an entity/object to keep and what to ignore" (Angeli et al., 2016), and as the practice of Abstracting and modularizing, that is "building something large by putting together collections of smaller parts" (Brennan & Resnick, 2012).

The analysis of the reviewed studies reveals the following **CT practices** according to Brennan & Resnick's (2012) framework: Testing and Debugging, Remixing and Reusing code, Being incremental and iterative, Abstracting and Modularizing. In addition, elements such as Design for usability, Code organization and documentation, and Programming efficiency proposed by Denner et al. (2012) as key competences for engaging in CT are also evident.

CT concepts as defined by Brennan & Resnick (2012) that repeatedly arouse in the examined studies are Sequences, Conditionals, Loops, Events, Parallelism, Variables (Data), and Operators. Functions, Synchronization blocks and User Interactivity blocks that are not included in Brennan & Resnick's (2012) framework, are also evident. Researchers (e.g., Moreno León et al., 2015; von Wangenheim et al., 2018) in the reviewed empirical studies often match these concepts with other CT elements. For example, von Wangenheim et al. (2018) assign abstraction to the use of more than one script and the definition of custom blocks in Snap!.

The examination of the studies also reveals the presence of elements such as Logic, Collaboration, Cooperativity, Problem solving, Creativity, Communication, Critical Thinking, Self-efficacy and others that appear once or twice and are not included in CT frameworks. The presence of these elements could be explained since some validated general assessment methods such as Dr. Scratch (Moreno León et al., 2015) and CTS (Korkmaz, Çakir, & Özden, 2017) assess these skills. These general methods are adopted by other studies (Durak, Yilmaz, & Bartin, 2019; Gabriele et al., 2019; Garneli & Chorianopoulos, 2018, Garneli & Chorianopoulos 2019; Günbatar, 2019; Korkmaz & Bai, 2019; Marcelino, Pessoa, Vieira, Salvador, & Mendes, 2018), resulting in a strong presence of these elements in the reviewed empirical studies.

CT attitudes and perspectives appear less frequently in the reviewed studies and include mainly Connecting and Expressing as described by Brennan & Resnick (2012).

Table 5

Knowledge Base sub-areas

CT elements classification	Description	CT frameworks
Concepts	Concepts (programming elements) encountered during programming.	Brennan & Resnick (2012)
Skills	The ability and capacity to carry out CT thought processes.	CSTA & ISTE (2011), Angeli et al. (2016), Shute et al. (2017)
Practices	Thinking and learning processes developed during programming.	Brennan & Resnick (2012)
Perspectives	Perception of oneself, his/her relationship with others and the digital world.	Brennan & Resnick (2012)
Attitudes	Dispositions and mindsets.	CSTA & ISTE (2011), Barr & Stephenson

(2011)

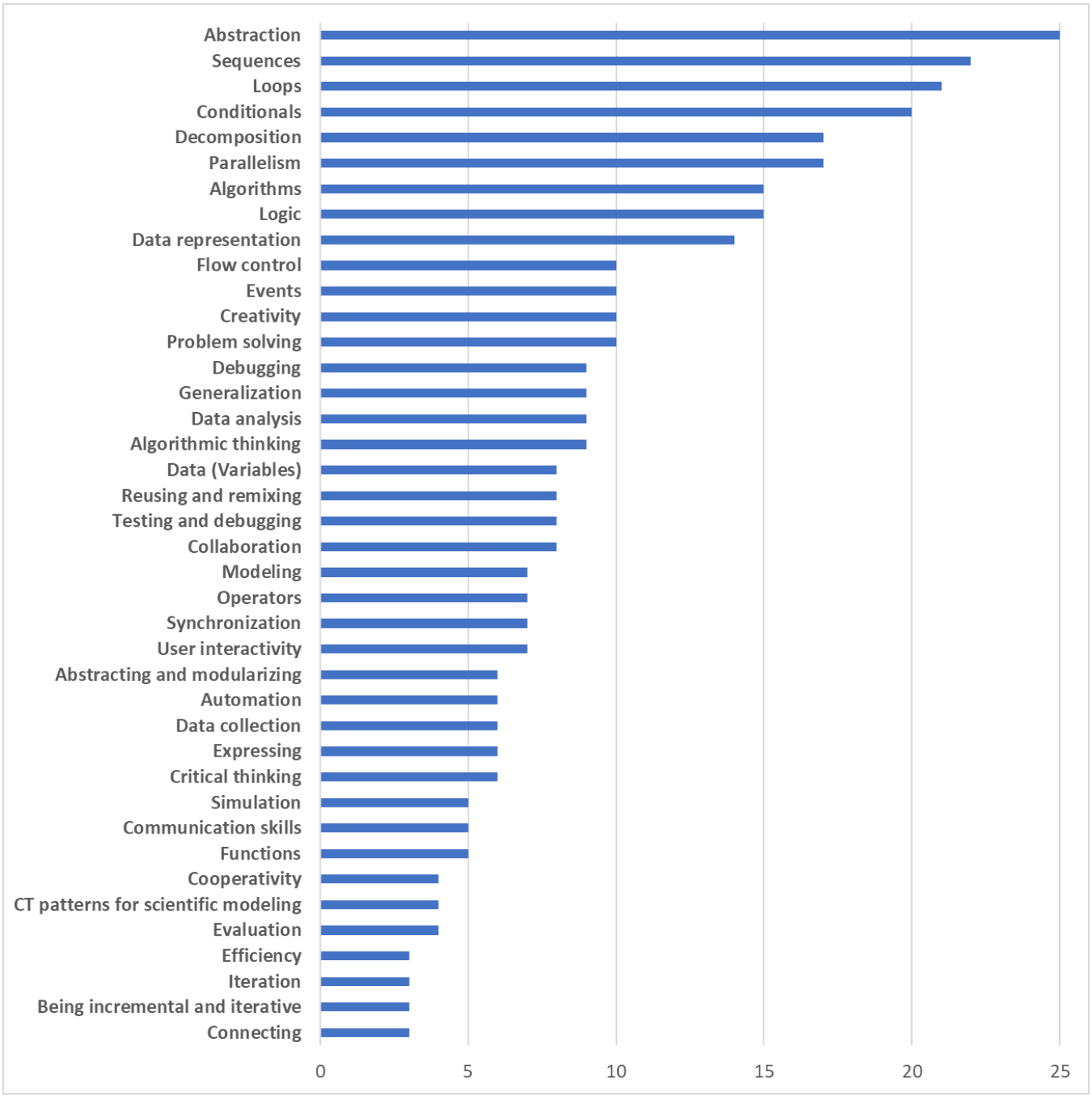


Fig. 6. Number of studies by CT element appearing more than twice in the examined studies.

4.2.2 Assessment Area

CT assessment is examined in 53 studies. Researchers in the examined studies develop and validate assessment methods, propose frameworks or measure students' CT in order to achieve deep understanding of students' learning (Fronza et al., 2017) through various assessment methods. We classify Assessment Area into five sub-areas: Self-report methods; Tests; Artifact Analysis; Observations; and Frameworks. Tests, Artifact Analysis and Observations measure directly CT, in contrary with self-report methods that measure CT indirectly through recording self-reflection. Table 6 presents the classification of Assessment.

CT assessment methods in the examined studies are mainly based on the specific content of each study. However, there are some efforts to develop general assessment methods. These efforts include development and validation of tests (Chen et al., 2017; Román-González et al., 2017), self-report scales (Kong, Chiu, & Lai, 2018; Korkmaz et al.,

2017; Kukul & Karataş, 2019; Yağcı, 2019) for general use in CT assessment and automatic artifact analysis instruments (Moreno León et al., 2015). Artifact analysis involves examining students' programs to detect evidence of CT. Automatic artifact analysis allows teachers and researchers to focus on assessment methods such as observations and interviews to gain a complete picture of students' understanding (Da Cruz Alves et al., 2019).

Assessment frameworks usually propose optimal combinations of assessment methods. Frameworks that have been proposed involve data mining techniques (De Souza, Barcelos, Munoz, Villarroel, & Silva, 2019), hypothesis-driven approaches (Grover et al., 2017) and Evidence-Centered-Design (ECD) methods (Snow, Rutstein, Basu, Bienkowski, & Everson, 2019).

Table 6

Assessment sub-areas

			Studies
Indirect Methods	**Self-Report Methods**	scales, questionnaires, surveys, interviews, think-aloud protocol, journals, reflection reports	S2,S4,S6,S12,S13,S18,S30,S35,S36,S39,S40,S47,S48,S55,S56,S57,S59,S60,S61,S66,S70,S79,S88,S95,S97,S101
Direct Methods	**Tests**	multiple-choice tests, quizzes, open-ended and other tasks, tasks and assignments with rubrics, semi-finished programs, projects, design scenarios	S3,S6,S9,S10,S11,S12,S13,S15,S19,S32,S39,S53,S70,S75,S76,S77,S79,S84,S85,S90,S93,S95,S100,S101
	Artifact analysis	automatic analysis, manually inspection of artifacts for CT evidence, examination of artifacts for CT patterns, log data	S4,S10,S13,S15,S25,S26,S32,S33,S35,S36,S37,S44,S46,S54,S63,S65,S66,S72,S86,S88
	Observations	observations of students' actions, screen recordings, learning analytics, camera recordings, researchers' notes, structure-based observations	S4,S6,S10,S12,S35,S37,S70,S79
Frameworks	frameworks for CT assessment		S4,S10,S15,S25,S32,S37,S39,S84,S90,S101

4.2.3 Learning Strategies Area

Learning strategies are mentioned in 37 studies. We classify the most common learning strategies in six sub-areas: Game Based Related Strategies, Modeling & Simulations Based Related Strategies, Problem Solving Related Strategies, Project Based Related Strategies, Scaffolding Related Strategies and Collaborative Related Strategies (Table 7). Scaffolding Related Strategies are classified as a separate sub-area, as they are particularly emphasized in the selected studies. Other strategies involve hands-on, aesthetic design through media design, storytelling and guided-discovery. Fig. 7 presents the number of CT studies by most common strategies.

Studies focusing on learning strategies either propose a pedagogical framework for CT or apply learning strategies to motivate students and enable them acquire CT. Many of these strategies are linked to constructionism (Papert, 1980) grounded in Piaget's (1970) constructivist theory, and/or Vygotsky's (1978) Zone of Proximal Development. Additionally, learning strategies are implemented in traditional classroom settings, at distance or in blended environments (e.g., Basogain et al., 2018; Grover et al., 2015) that take advantage of the presence of teachers and the services provided by virtual learning environments. Researchers in selected studies often use multiple learning strategies to take advantage of their benefits. Out of the 37 studies included in this CT Area, 15 apply or propose more than one learning strategy.

Table 7

Learning strategies sub-areas

		Studies
Game Based Related Strategies	Game Based Related Strategies involve game design and digital/video game development, programming games and any strategy that exploits games and programming.	S4,S25,S26,S35, S36,S46,S48,S5 3,S60,S72,S89, S100
Modeling & Simulations Based Related Strategies	Modeling & Simulations Based Related Strategies involve designing of scientific models and simulations through strategies such as scientific inquiry and learning by design.	S2,S11,S28,S35, S72,S81
Problem Solving Related Strategies	Problem Solving Related Strategies involve Problem Based Learning and problem-solving learning strategies in general.	S5,S39,S51
Project Based Related Strategies	Project Based Related Strategies involve the engagement with authentic projects set around real challenges and problems.	S32,S53,S69,S7 0,S79
Scaffolding Related Strategies	Scaffolding Related Strategies involve strategies that offer support to students as they learn, including instructional scaffolding, support/guidance, and adaptive, peer-, resource- scaffolding.	S6,S11,S13,S17, S26,S36,S39,S4 5,S70,S72,S81,S 93
Collaborative Related Strategies	Collaborative Related Strategies involve strategies where students actively interact during the learning process including collaborative learning, teamwork, pair programming and strategies based on student's collaboration.	S6,S30,S33,S45, S48,S70

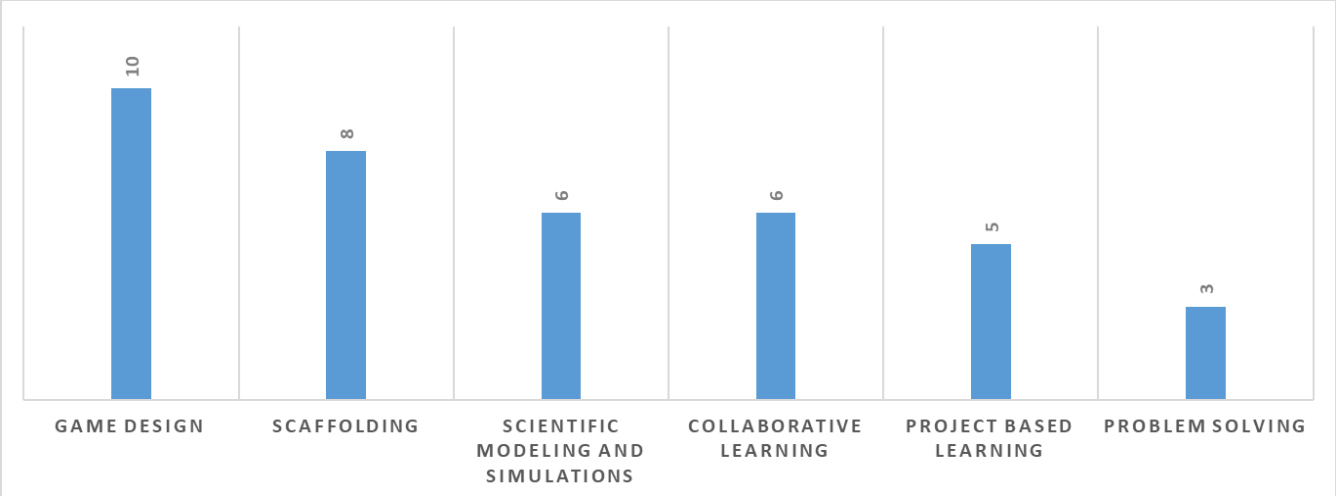


Fig. 7. Number of studies by the most common learning strategies.

4.2.4 Factors Area

CT-related factors are discussed in 22 studies. We classify Factors Area in two sub-areas: Demographic factors and Cognitive & non-cognitive factors presented in (Table 8). Demographic factors have the strongest presence in the selected studies with gender being discussed in 17 and grade level in 7 out of 22 examined studies.

Several studies investigate the relationship between CT and grade level. Some of them (Atmatzidou & Demetriadis, 2016; Werner, Denner, Campe, & Kawamoto, 2012) conclude that CT acquistition is not grade-related (or age-

related). Several other studies conclude that there is a significant relationship. However their results on the type of this relationship are contratictory. On the one hand, some studies conclude that there is a positive relationship between grade level or age and CT. More specifically, Román-González, Pérez-González, & Jiménez-Fernández (2017) assessed 1,251 students' CT using Computational Thinking Test (CTt). They concluded that CT levels increased with the grade, thus suggesting that this finding may be related to the cognitive problem-solving aspect of CT. This finding is in line with the results reported by Durak et al. (2019). On the other hand, there are studies providing evidence that there is a negative relationship between age (grade level) and CT. More specifically, Durak & Saritepeci (2018) found that grade level negatively predicted CT, suggesting that as the students' grade level increases their CT levels are negatively affected. However, they note that this finding may be related to participants' prior experience, which was different depending on the grade level. A negative relationship between CT (elements of programming empowerment) and grade level has also been reported in Kong, Chiu, & Lai's, (2018). However, authors emphasize that other factors such as less personalized instruction and differences in the level of difficulty may have affected students' CT acquisition. Israel-Fishelson & Hershkovitz (2019) go further by comparing students' achievement in specific CT elements between their different grade levels. The authors emphasize that students at different grade levels performed better on different concepts, suggesting that the design of a CT approach should take into account "the fit between CT concepts and grade level" (Israel-Fishelson & Hershkovitz, 2019).

Studies that investigate gender relationship with CT are also contradictory. Some of them conclude that learning CT is gender-related, while others (Atmatzidou & Demetriadis, 2016; Werner et al., 2012) find that there is no significant relationship between gender and CT learning. Studies that conclude that CT is gender-related are also contradictory. Some of them (e.g., Durak & Saritepeci, 2018; Durak et al., 2019) found CT level differentiation in favor of female while others (e.g., Kong et al., 2018; Román-González et al., 2017) in favor of male students. Studies (e.g., Cooper et al., 2014; Fletcher & Lu, 2009; Repenning et al., 2015) also discuss challenges related to demographic factors (e.g., gender, socio-economic) such as underrepresentation in CS and students' low motivation.

Creativity appears in the selected studies in the light of two different perspectives. Several studies (Allsop, 2019; Kim & Kim, 2016; Korkmaz et al., 2017; Yağcı, 2019; Zhong et al., 2016) place creativity in the core of CT along with other elements. However, other studies approach creativity as a separate construct and examine its relationship to CT. Teachers who participated in Nouri, Zhang, Mannila, & Norén (2019) reported creativity as one of the skills occurred during CT learning. Kim & Kim (2016) found that students' creativity was improved after they participated in their CT intervention. On the contrary, Hershkovitz et al. (2019) found no relationship between CT and creativity. However, they suggest that this may relate to specific features of the learning platform used.

Self-efficacy is an additional factor that appears in the selected studies in the light of the two aforementioned perspectives. Román-González, Pérez-González, Moreno-León, & Robles (2018) found that CT was positively related to CT self-efficacy. In addition, they suggested that fostering students' self-efficacy through positive and personal learning experiences might be effective in acquiring CT. A significant relationship between CT and programming self-efficacy was also reported by Durak et al. (2019).

Other factors addressed in the selected studies include aspects of personality (Román-González et al., 2018), persistence (Israel-Fishelson & Hershkovitz, 2019), attitudes toward and interest in programming, (Kong et al., 2018; Witherspoon & Schunn, 2019) attitudes toward collaboration (Kong et al., 2018), academic success and attitude against various school subjects (Durak & Saritepeci, 2018), challenges in learning programming (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013) and teachers' instructional goals (Witherspoon & Schunn, 2019).

Cognitive factors such as verbal, spatial, reasoning, numerical and problem-solving ability (Román-González et al., 2017), ways of thinking (Durak & Saritepeci, 2018), and reflective thinking (Durak et al., 2019) are also investigated in the literature.

Table 8

Factors sub-areas

		Studies
Demographic factors	Grade level, gender, socio-economic and cultural background	S4,S6,S22,S29,S30,S31,S43,S45,S49,S53,S55,S56,S70,S72,S76,S77,S90

Non-Cognitive and Cognitive factors	Personal traits, attitudes and motivations such as aspects of personality, creativity, self-efficacy, persistence, attitudes toward programming and attitudes toward collaboration; academic performance, challenges in learning programming Factors that involve cognitive functions and mental abilities such as verbal, spatial, reasoning & numerical ability and problem-solving ability	S29,S30,S42,S46,S55,S66 ,S76,S77,S81,S90, S93

4.2.5 Tools Area

Researchers in 47 studies use or develop tools for CT teaching and learning. We classify tools leveraged for teaching and learning CT through programming in K-12 education in three sub-areas: programming tools & communities, robotics & microcontrollers, and tools specifically developed for CT. Table 9 presents the classification of tools. Fig. 8 presents the number of studies by tool.

Students in the selected studies are mainly engaged with programming concepts and practices through programming tools. According to Brennan & Resnick (2012) the concepts and practices that students encounter during programming could be considered as CT concepts and practices as well. Most of the tools recorded in the selected studies are visual programming tools. Furthermore, even when text programming is used, the outcome of programming is often visualized through animations. Agent-based programming paradigm is also widely applied. In addition, communities are proposed by authors (e.g., Clark & Sengupta, 2019; Kafai, 2016) who argue that CT and programming are social practices. Students in the selected studies share their programs and use socialization features of communities that according to Xing (2019) can lead to CT development.

Robotics are used for teaching and learning CT in some of the selected studies. Students in these studies encounter CT concepts and practices during programming robots to interact with the environment. Among other tools educational robotics kits have the strongest presence (e.g, Atmatzidou & Demetriadis, 2016; Chalmers, 2018). Microcontrollers are also evident in studies (e.g., Carlborg, Tyrén, Heath, & Eriksson, 2019; Durak et al., 2019) where students program automations or complex robotic devices.

Several studies develop tools in order to support a CT theoretical framework or curriculum. Most of the developed tools are visual programming tools and involve game play (e.g. Clark & Sengupta, 2019; Weintrop, Holbert, Horn, & Wilensky, 2016) and/or modeling (e.g. Basu, Biswas, & Kinnebrew, 2017; Clark & Sengupta, 2019; Kynigos & Grizioti, 2018; Sengupta et al., 2013).

Table 9

Tools sub-areas

		Studies
Programming tools & Communities	Visual & text programming tools. Communities that provide users with the opportunity to interact with other programmers.	S2,S4,S5,S10,S15,21,S26,S30,S 32,S33,S35,S36,S37,S39,S42,S 44,S45,S46,S48,S49,S53,S54,S 58,S60,S63,S70,S71,S72,S75,S 79,S86,S94,S101
Robotics & Microcontrollers	Programmable robot constructs including educational robotics kits, physical & virtual robots. Automations, control devices, interactive physical systems.	S6,S12,S13,S17,S18,S19,S30,S 60,S93
Tools specifically developed for CT	Tools developed to support a CT theoretical framework or curriculum.	S11,S21,S47,S59,S81,S89, S93, S100

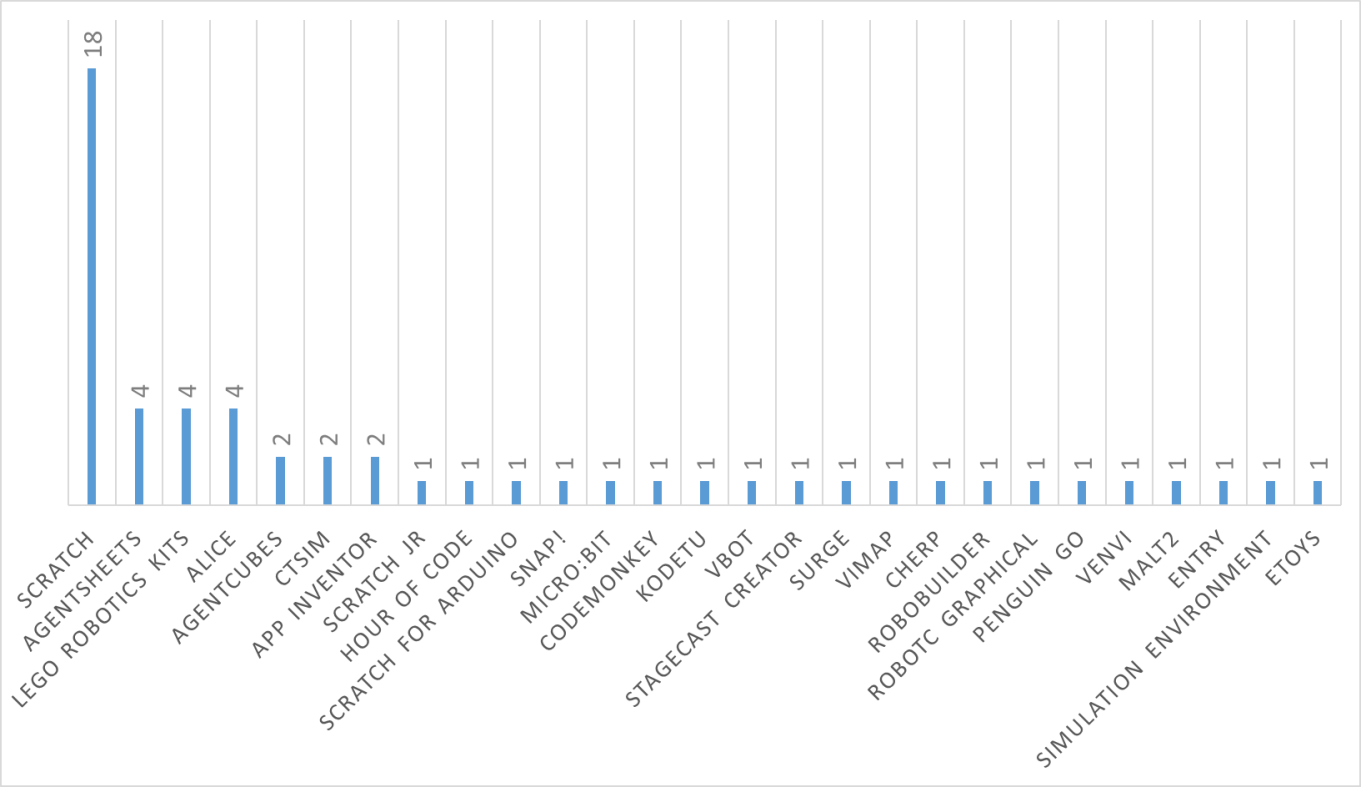


Fig. 8. Number of studies by tool.

4.2.6 Capacity Building Area

Providing guidance and support to teachers is discussed in 19 studies. We classify Capacity Building Area in three sub-areas: Knowledge for teaching CT, Teacher Education and Professional Development (Table 10).

The specification of knowledge for teaching CT is a prerequisite for teacher support (Angeli et al., 2016; Cooper et al., 2014) and thus, we classify it as a separate sub-area in Capacity Building Area. Technological Pedagogical Content Knowledge (TPCK or TPACK) is proposed for specifying this knowledge in the selected studies (e.g., Angeli et al., 2016; Mouza, Yang, Pan, Yilmaz Ozden, & Pollock, 2017). TPCK interweaves the knowledge of technology (TK), content (CK) and pedagogy (PK) (Koehler & Mishra, 2006). Angeli et al. (2016) define TPCK for CT as the knowledge that enables teachers to identify creative and authentic CT projects; identify technologies that provide the necessary technological means for practicing/teaching the whole range of CT; and use representations in order to make CT comprehensible for all. Other researchers (e.g. Mouza et al. 2017) place CT into the Technology Knowledge (TK), suggesting that teachers should understand this knowledge and draw connections with PK and disciplinary content (CK), such as math, language, art.

Teacher Education could be based on revised educational technology courses that provide pre-service teachers with CT opportunities and methods courses that focus on teaching and learning and facilitate the integration of CT into pre-service teachers' future educational practices (Yadav et al., 2017). Along these lines, studies in this sub-area introduce CT to pre-service teachers through technology courses (Angeli et al. 2016, Gabriele et al., 2019; Mouza et al., 2017; Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014) and methods courses. For example, Adler & Kim (2018) incorporated CT into a science methods course for pre-service teachers. A high percentage of participants (90%) who engaged with CT through simulations consider that CT and simulations could be integrated into the classroom environment. Participants in Gabriele et al. 's (2019) study developed projects in Scratch and subsequently incorporated them into their teaching practices during their internship.

Professional Development aims to support teachers in understanding and integrating CT into their practices (Alfayez & Lambert, 2019; Bower, Wood, Lai, Howe, & Lister, 2017). Hickmott & Prieto-Rodriguez (2018) propose that Professional Development should (a) provide activities relevant to both CT tools and CT learning strategies; (b) include both step-by-step exercises and self-directed projects; (c) take into account teachers' prior knowledge; (d) provide resources that can be directly integrated into teaching practices; and (e) assess teachers' knowledge acquisition through direct assessment methods. Kale et al. (2018) argue that when Professional Development focuses on the application of CT in different domains and problem solving, it allows teachers to recognize the

importance of CT and integrate the knowledge gained into their teaching. Ongoing professional development that involves workshops, embedded coaching, administrative support, co-planning lessons and co-teaching, could also provide inservice teachers with valuable assistance and thereby expanding their participation in CT (Israel, Pearson, Tapia, Wherfel, & Reese, 2015).

Table 10

Capacity Building sub-areas

		Studies
Knowledge for teaching CT	Models for specifying the knowledge that teachers need for teaching CT.	S5,S18,S22,S67,S96
Teacher education	Undergraduate courses such as educational technology and methods courses that promote CT learning and teaching.	S2,S33,S34,S67,S95,S96
Professional development	Variety of tools such as workshops, training, courses designed to help teachers improve their professional knowledge.	S8,S14,S18,S41,S44,S45 ,S50,S61,S63,S69,S82

5. Computational Thinking through Programming in K-12 education (CTPK-12) model

The proposed Computational Thinking through Programming in K-12 education (CPTK-12) conceptual model (Fig. 9) is based on the extracted CT Areas (as described in Section 4.2) and their relationships presented in this section (Table 11). CTPK-12 models' relationships show the dominant relations between CT Areas as they emerge from the selected studies.

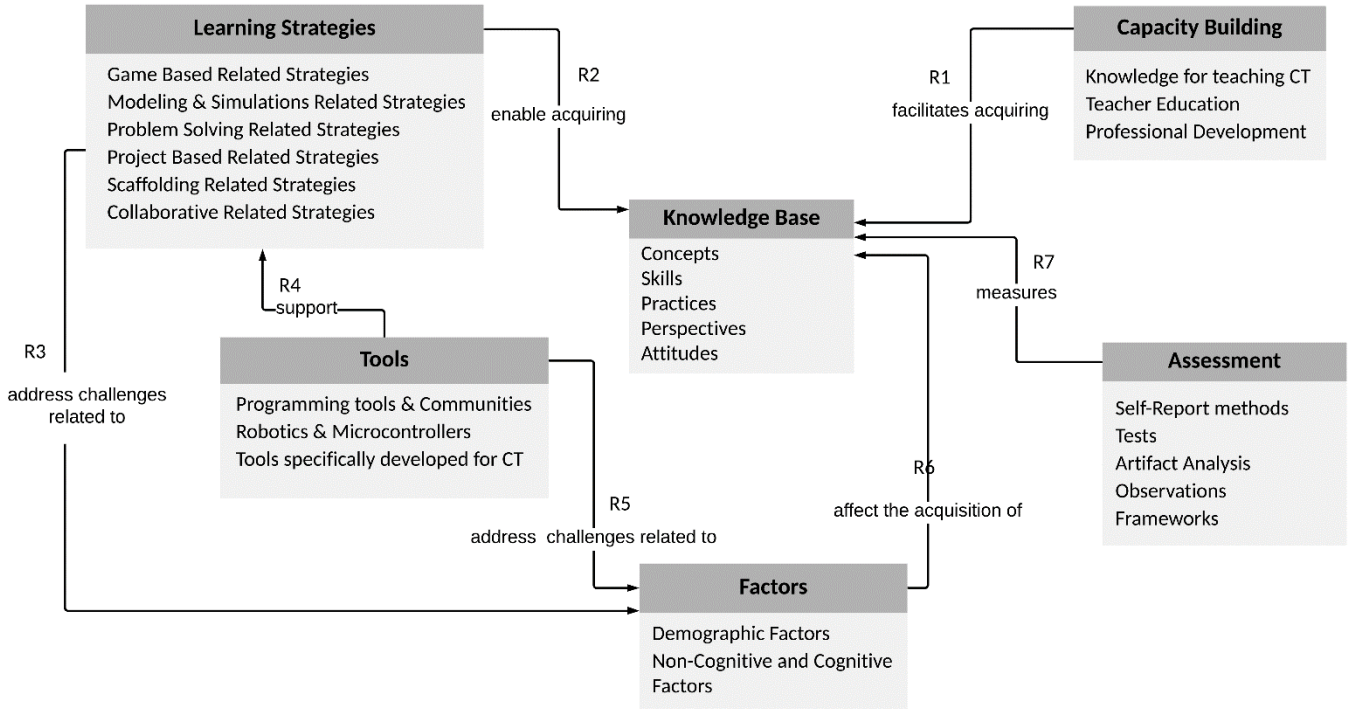


Fig. 9. Computational Thinking through Programming in K-12 education (CTPK-12) model.

Table 11

CT Areas' relationships

Capacity Building	R1. Supports teachers to facilitate students understand and acquire CT.

Learning Strategies	R2. Enable students understand and acquire CT. R3. Increase the motivational levels of underrepresented students, thereby broadening CT participation and addressing underrepresentation due to socio-economic, cultural and gender differences.
Tools	R4. Allow students to acquire CT through supporting learning strategies. R5. Address the challenges encountered in learning programming and reinforce underrepresented students' motivation.
Factors	R6. Affect the acquisition of CT.
Assessment	R7. Measures CT and provides a means for deep understanding of students' learning.

R6 and **R7** model's relationships could be considered plausible and are widely reflected in the studies included in the respective CT Areas described in section (4.2). The same is true for **R1**, while this relationship is not widely tested empirically in the selected studies. The remaining relationships are further elaborated in this section.

R2. Several studies attribute the success of the proposed interventions to the applied strategies. Grover et al. (2015) place particular emphasis on the pedagogical design of their strategy, which eventually led to the students' understanding of CT concepts (algorithmic constructs). Repenning et al. (2015) also found that Scalable Game Design strategy that involves game design, simulations and scaffolding allowed students to develop CT skills, highlighting the important role of pedagogy in the strategy. Sáez-López, Román-González, & Vázquez-Cano (2016) implemented an active pedagogical approach, concluding that primary school students who participated in their study improved their CT levels in regard to CT concepts, logic and CT practices. In addition, there are also findings that support the assumption that learning strategies such as Game Design (Garneli & Chorianopoulos, 2019), Project Based Learning enhanced with software agile methods (Fronza et al., 2017) and Modeling & Simulations (Garneli & Chorianopoulos, 2018) enable the acquisition of CT.

R3. Studies also discuss the role of learning strategies in relation to challenges posed by demographic factors (e.g. gender, socio-economic background) such as underrepresentation in CS and students' low motivation (Cooper et al., 2014; Fletcher & Lu, 2009), arguing that CT teaching and learning motivates learners, especially females and underrepresented students. More specifically, Ioannidou, Bennett, Repenning, Koh, & Basawapatna (2011) and Repenning et al. (2015) suggest that Scalable Game Design learning strategy leads in broadening participation in CS. Out of over 4000 students who participated in Scalable Game Design Project, 56 % were minority students and 45% were female. 64% of the participated girls were interested in continuing their CT activities. In addition, ethnic minority factor did not affect students' interest in continuing involving with CT (Repenning et al., 2015). Teachers who participated in Israel's et al. (2015) study, used teaching CT through collaborative problem solving, modeling, explicit instruction, peer collaboration, and guided discovery in order to make CS accessible to students with low financial backgrounds and disabilities.

R4. Learning strategies are supported by tools. Out of 32 empirical student-centered studies, 21 utilize tools as a means of supporting learning strategies to introduce students to CT. Specific features of tools could support different learning strategies. For example, a strategy that involves modeling is supported among others, by tools that include a modeling environment such as CTSiM (Basu et al., 2017; Sengupta et al., 2013). A game design strategy is often supported in the selected studies by tools such as Scratch (Resnick et al., 2009) that allows students of all ages to develop games through its low floor environment.

Furthermore, there is evidence that engaging with tools without a learning strategy is not enough to gain knowledge of CT. Denner et al. (2012) analyze 108 games created by middle school students in Creator, finding lack of code organization, documentation and design for usability. Since they found that participated students faced challenges in designing their games and understanding several programming concepts, they suggested that proper guidance is critical to enable students' motivation. Brennan & Resnick (2012) noted that interviewee students that developed projects in Scratch, sometimes could not explain their programs, although they had incorporated several programming constructs into them. Zhao & Shute (2019) examined the development of students' CT through a game environment they developed, noting that a non-trivial part of the students' improvement in CT could be attributed to increased familiarity with the environment.

R5. There is also evidence that tools enhance underrepresented students' engagement in programming and CS. In a

study by Kim & Kim (2016), participating elementary female students reduced their negative attitudes towards software education after following a CT course and designing games in App Inventor.

In addition, several studies emphasize (e.g., Fronza et al., 2017; García-Peñalvo & Mendes, 2018; Lye & Koh, 2014; Repenning, Basawapatna, & Escherle, 2017; Sengupta et al., 2013) that certain tool features (e.g visual interfaces) eliminate the challenges related to the nature of programming, such as difficulty of learning a complex programming syntax.

6. Discussion

The analysis of Knowledge Base Area reveals that recent years' efforts to identify measurable elements of CT have led to various terms describing classifications of CT elements such as concepts, practices, skills, attitudes, perspectives. These terms are often presented with different meaning. In addition, several CT elements proposed by frameworks appear to be classified in various ways. For example, abstraction occurs as the thought process of "the ability to think in abstractions" (Selby, 2013), as the skill "to decide what information about an entity/object to keep and what to ignore" (Angeli et al., 2016), and as the practice of Abstracting and modularizing, that is "building something large by putting together collections of smaller parts" (Brennan & Resnick, 2012).

During the analysis of the studies we recorded more than 60 different CT elements proposed by frameworks and definitions or simply assessed in empirical studies. Some of them are not included in definition frameworks. This could be explained by the evolution of the domain. As research in the domain progresses, empirical studies introduce further CT elements in their assessments in addition to those proposed by the respective frameworks. The strong presence of some of these elements in the reviewed studies is due to the fact that they are included in assessment methods such as Dr. Scratch (Moreno León et al., 2015) and CTS (Korkmaz et al., 2017) that have been adopted by other studies (e.g. Durak, Yilmaz, & Bartin, 2019; Gabriele et al., 2019; Garneli & Chorianopoulos, 2018, Garneli & Chorianopoulos 2019; Günbatar, 2019; Korkmaz & Bai, 2019; Marcelino, Pessoa, Vieira, Salvador, & Mendes, 2018).

Many of the reviewed empirical studies assess CT as a skill. This could be explained, since CT was introduced as a skill and attitude by the widely accepted definition of Wing (2006). In addition, the term CT skills emerges from definitions and frameworks such as (Angeli et al., 2016) and (CSTA & ISTE, 2011). Programming constructs or CT concepts as described by Brennan & Resnick (2012) are also frequently assessed. This finding is consistent with the results presented by Zhang & Nouri (2019). This is likely because CT concepts can be assessed by direct assessment methods and in addition some of these methods provide automation facilitating the assessment process. On the contrary, it is likely that the difficulty to assess perspectives and attitudes through direct assessment methods leads to its low presence in the reviewed studies.

CT assessment methods mainly assess CT through pretest/posttest, self-report and artifact analysis. In order to gain a complete picture of the learning process, several studies include observations in their assessment. CT assessment methods are mainly based on the specific content of each study although there are some efforts to develop assessment methods for general use. Most of these methods are self-report methods assessing CT indirectly, proposing CT elements that are absent in definition models. Thus, we can conclude that there is no agreement on what and how to assess CT. This is consistent with several studies (Brennan & Resnick, 2012; Denning, 2017; Fronza et al., 2017; Grover et al., 2017, Grover et al., 2015; Moreno León et al., 2015; Werner et al., 2012; Zhong et al., 2016) that highlight the challenge of CT assessment.

The examination of the studies also reveals that the most common proposed learning strategies are Game Based Related Strategies and Modeling & Simulations Related Strategies leveraging scaffolding and collaborative strategies. This could be explained as game design increases the motivational level of students while modeling & and simulations facilitates processes that are core to CT such as Abstraction and Evaluation. There is evidence that learning strategies that enhance students' CT learning are essential, as there is research that reveals that introducing CT to young students without considering appropriate learning strategies leads to difficulties for students to acquire CT.

Tools in the reviewed studies provide environments and communities where students are engaged with programming constructs and practices. Most of them share the common feature of visual programming. Scratch is the most commonly used tool and is usually used for game and media design. This is likely due to the combination of the following reasons: a) Scratch is proposed as a tool to support CT development by its designers (Resnick et al., 2009), b) Brennan & Resnick's (2012) framework in which CT elements are defined in relation with Scratch, facilitates researchers to use Scratch in their studies and c) the assessment of CT through projects developed in Scratch is

facilitated by automatic assessment methods such as Dr. Scratch (Moreno León et al., 2015).

Several studies examine CT-related factors including cognitive, non-cognitive and demographic factors. Determining the relationship between these factors and CT could indicate the most appropriate approaches for each case depending on the presence of these factors. Most of the studies examine gender and socio-economic factors and challenges that arise from them such as students' underrepresentation and gender and social differences. The examination of the selected studies indicates that while factors may affect CT development, teaching and learning CT could address low enrollment in CS and increase interest of underrepresented students. Researchers and teachers in the examined studies are not particularly concerned about challenges that could affect CT acquisition due to the nature of programming as discussed in (Buitrago Flórez et al., 2017). This could be explained as the tools used have features that eliminate these difficulties.

Capacity Building has gained attention especially after 2015. Teacher education, professional development and the knowledge that teachers need in order to teach CT are the main issues discussed in the selected studies. Many of these studies are surveys that examine the challenges faced by teachers. Other studies propose frameworks or discuss professional development and teacher education interventions.

The proposed CTPK-12 conceptual model is developed to aid domain understanding, communicate domain details and document CT through programming in K-12 domain for future reference. The CTPK-12 conceptual model can be expanded to include higher education or other approaches than programming, such as kinesthetic approaches. Thus, it has the potential to serve as a basis for future studies by including CT Areas or sub-areas as the domain evolves.

In addition, the CTPK-12 model could serve as a basis for hypothesized research models that establish a direct link between theory and statistical estimations. An example is presented in (Fig. 10) where research hypothesis is developed between some CT Areas of the model. Research hypothesis in the specific example includes H1 (Between Learning Strategies Area and Knowledge Base Area): Game design enables the acquisition of CT skills. H2 (Between Learning Strategies Area and Factors Area): Game design motivates female students, addressing gender differences. H3 (Between Tools and Learning Strategies): Scratch provides opportunities for game development, supporting game design. H4 (Between Tools and Factors): Scratch motivates female students, addressing gender differences. H5 (Between Factors and Knowledge Base): Female and male students acquire a different level of CT skills.

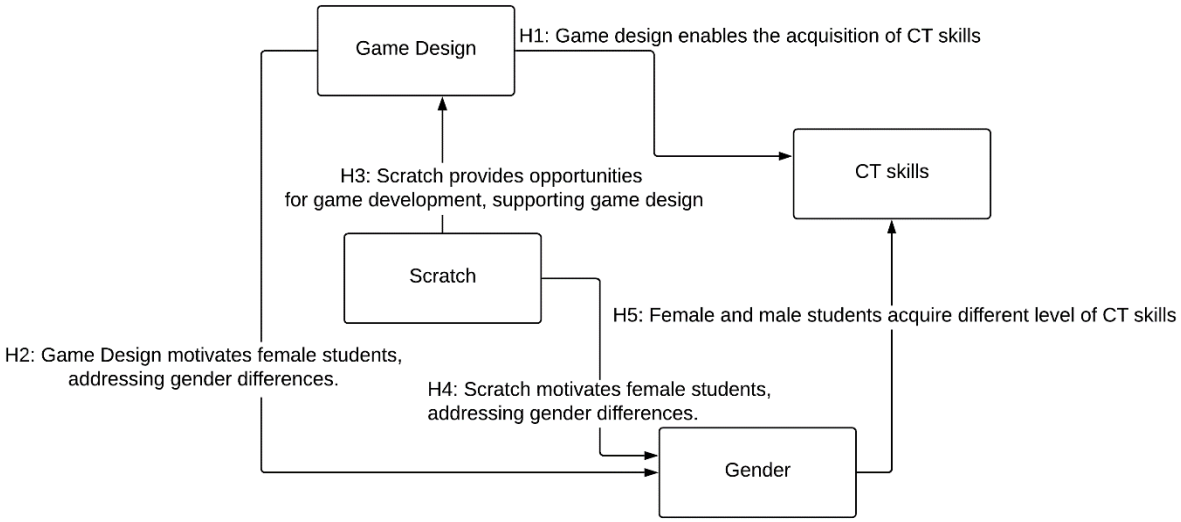


Fig. 10. Example of a hypothesized research model based on CTPK-12 model.

We suggest using the CTPK-12 conceptual model to design empirical interventions aimed at teaching and learning CT through programming in K-12 education to investigate as many CT Areas as possible. Furthermore, we assert that empirical studies that explicitly define the targeted elements of the CT knowledge base, the learning strategies applied, the assessment methods used, the tools used, the factors that may affect CT based on the profile of participants, and the capacity building of teachers involved, provide a complete picture of the intervention being attempted.

In addition, the CTPK-12 conceptual model could be combined with models for CT activities such as the scope of autonomy model (Carlborg et al., 2019) and the constructionism matrix (Csizmadia, Standl, & Waite, 2019). The CTPK-12 model could be used as a guide to designing teachers' lessons, providing them with evidence-based results

and detailed information on CT through programming in K-12 education and facilitating them to integrate CT into their educational practices. The models' areas and their relationships could be taken into account during designing of curricula as well as CT teaching and learning process to improve effectiveness. In addition, CTPK-12 model could inform policy makers on their decision-making regarding CT and integration into K-12 education. It should be noted that the application of the CTPK-12 model in practice should take into account the settings under which CT will be incorporated. These settings include parameters such as course type (optional or compulsory) or whether CT will be employed into other courses in the curriculum or as a separate course. Further elaboration of these settings is outside the scope of this study. Fig. 11 presents the possible application of CTPK-12 model in educational practice.

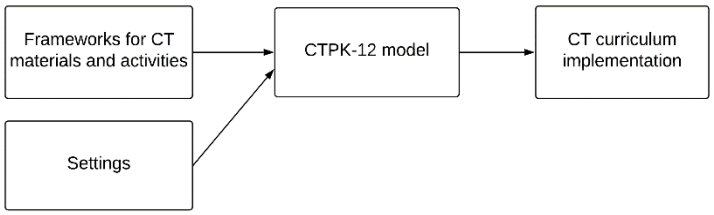


Fig.11. CTPK-12 model application in K-12 educational practice.

7. Conclusions and future research directions

In this study, a conceptual model of CT through programming in K-12 education (CTPK-12) was developed. The proposed model is based on a systematic literature review and the identification of CT Areas and their relationships. CT Areas result from the recording of all topics of interest to researchers, as discussed in the scientific publications. CTPK-12 model provides an overall map of the domain that aids domain understanding and could serve as a basis for future studies and facilitate the integration of CT into K-12 educational practices.

The CTPK-12 model indicates that CT through programming in K-12 education domain includes the following six areas Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building area that are related to each other. Some of the relationships between the areas have not yet been sufficiently explored including (a) which tools support which learning strategies, (b) which learning strategies enable the acquisition of CT, (c) which factors affect CT development and (d) how capacity building affects students' CT levels.

The CTPK-12 model also reveals that although the focus on Assessment, Tools and Factors area remains approximately constant over time, it increases for Learning Strategies and Capacity Building area and decreases for Knowledge Base area. This marks a change in the focus of research that could be interpreted as a shift to more tangible issues of educational practice. The findings also indicate gaps and future directions regarding the models' areas and relationships that are presented in the following paragraphs.

Assessment area is at the forefront of CT research gathering the greatest interest of researchers in the selected studies. However, CT assessment methods in the selected studies include mostly methods based on particular activities and curricula and therefore their use in different contexts is difficult. Efforts have been made to develop validated methods for general use that allow researchers to document their results based on validated instruments. Most of these methods are self-report methods; therefore, there is a need for additional validated methods, which could be applied to various settings, providing opportunities to standardize the CT assessment based on methods other than self-report.

Tools area is also one of the major topics investigated in the selected studies. Several studies focus on the development of environments designed specifically to support CT teaching and learning strategies. Although these environments are designed on the basis of CT frameworks, they are not yet widely used in empirical studies or educational practices aimed at developing CT. Instead, they appear only once in the literature in the studies where they are introduced. Therefore, beyond the theoretical basis and technicalities of CT tools, researchers need to consider issues of usability, student motivation, teacher facilitation through available resources and frameworks, and ease of assessment through built-in automated assessment methods. In addition, future studies should explore the relationship between Tools and CT development providing insights on which tools could better support which CT learning strategies.

Learning Strategies area has gained increasing interest in recent years. However, several of the studies reviewed simply refer to the learning strategies applied without further describing how they were implemented. Focusing on

learning strategies, presenting the relevant background and how they are implemented could support a more comprehensive picture of the conditions and context of the proposed CT interventions. Studies could also propose frameworks that support leveraging CT learning strategies. In addition, future studies could explore the relationship between learning strategies and CT development and provide insights on which learning strategies are most suitable for students to acquire which CT elements.

Capacity Building is highlighted as a critical Area of CT presence within educational settings and one of the rising areas in the domain research. Nevertheless, studies still argue that teachers face significant challenges in incorporating CT practices such as lack of technological infrastructure, lack of time for lesson plans and materials preparation and limited instructional time (Adler & Kim, 2018; Bargury et al., 2012; Israel et al., 2015; Ozturk, Dooley, & Welch, 2018; Sentance & Csizmadia, 2017). Most important, teachers have low levels of CT content knowledge (Alfayez & Lambert, 2019; Angeli et al., 2016; Bower et al., 2017; Israel et al., 2015; Kale, Akcaoglu, Cullen, & Goh, 2018) and knowledge about how to teach CT (Chalmers, 2018). Thus, more Capacity Building interventions and frameworks are needed to support inservice and pre-service teachers to successfully integrate CT into their teaching practices. In addition, the relationship between capacity building and CT development could be investigated in future studies.

Factors area has also been investigated in several of the selected studies. However, some of the results of the studies are contradictory, so it is unclear whether and to what extent these factors lead to higher or lower CT levels. As Angeli & Giannakos (2020) point out, how CT skills, such as abstraction, problem decomposition, and data structures, map to different abilities, grade level, disciplines, gender, and educational level is still missing from the literature. Further studies in this direction could build clarity about factors that may affect CT acquisition. With regard to how CT could be utilized to motivate underrepresented groups, there are few studies (e.g Kim & Kim, 2016; Leonard et al., 2018; Pinkard, Martin, & Erete, 2019) specifically aimed at motivating girls and underrepresented minorities. More studies are required to provide evidence of the relationship between factors, learning strategies and tools and provide insights on if and how learning strategies and tools could broaden participation in CT and address challenges related to factors.

Finally, future work could fully exploit the potential uses of the CTPK-12 model in both educational practice and research. This could include studying the application of the proposed model to teaching practices and curriculum design. In addition, future work could extend the proposed model to include higher education to develop a holistic approach covering CT teaching and learning from early years until graduation.

Appendix A. List of selected studies

S1 (CSTA), & (ISTE). (2011). Operational definition of computational thinking. Retrieved from https://www.iste.org/explore/Solutions/Computational-thinking-for-all

S2 Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, Vol. 23, pp. 1501–1514. https://doi.org/10.1007/s10639-017-9675-1

S3 Alfayez, A. A., & Lambert, J. (2019). Exploring Saudi Computer Science Teachers' Conceptual Mastery Level of Computational Thinking Skills. *Computers in the Schools*, Vol. 36, pp. 143–166. https://doi.org/10.1080/07380569.2019.1639593

S4 Allsop, Y. (2019 Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30–55. https://doi.org/10.1016/j.ijcci.2018.10.004

S5 Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology and Society*, Vol. 19, pp. 47–57. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85000838214&partnerID=40&md5=3f014c90dafb945e90c9552f5a6ef17f

S6 Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, Vol. 75, pp. 661–670. https://doi.org/10.1016/j.robot.2015.10.008

S7 Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *Proceedings - Frontiers in Education Conference, FIE*. https://doi.org/10.1109/FIE.2012.6462365

S8 Barr, V., & Stephenson, C. (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), 48–54. https://doi.org/10.1145/1929887.1929905

S9 Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 245–250. https://doi.org/10.1145/1953163.1953241

S10 Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior*, Vol. 80, pp. 412–419. https://doi.org/10.1016/j.chb.2017.04.058

S11 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*, Vol. 27, pp. 5–53. https://doi.org/10.1007/s11257-017-9187-0

S12 Berland, M., & Wilensky, U. (2015). Comparing Virtual and Physical Robotics Environments for Supporting Complex Systems and Computational Thinking. *Journal of Science Education and Technology*, Vol. 24, pp. 628–647. https://doi.org/10.1007/s10956-015-9552-x

S13 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*, Vol. 72, pp. 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

S14 Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, 42(3), 53–72. https://doi.org/10.14221/ajte.2017v42n3.4

S15 Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*, 1–25. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

S16 Buitrago Flórez, F., 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*, Vol. 87, pp. 834–860. https://doi.org/10.3102/0034654317710096

S17 Carlborg, N., Tyrén, M., Heath, C., & Eriksson, E. (2019). The scope of autonomy when teaching computational thinking in primary school. *International Journal of Child-Computer Interaction*, *21*, 130–139. https://doi.org/10.1016/j.ijcci.2019.06.005

S18 Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93–100. https://doi.org/10.1016/j.ijcci.2018.06.005

S19 Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, Vol. 109, pp. 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

S20 Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, Vol. 62, pp. 563–573. https://doi.org/10.1007/s11528-018-0292-7

S21 Clark, D. B., & Sengupta, P. (2019). Reconceptualizing games for integrating computational thinking and science as practice: collaborative agent-based disciplinarily-integrated games. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1636071

S22 Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). Education: A future for computing education research. *Communications of the ACM*, *57*(11), 34–36. https://doi.org/10.1145/2668899

S23 Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. Retrieved from Computing at Schools. website: https://community.computingatschool.org.uk/resources/2324/single

S24 Da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, Vol. 18, pp. 17–39. https://doi.org/10.15388/infedu.2019.02

S25 De Souza, A. A., Barcelos, T. S., Munoz, R., Villarroel, R., & Silva, L. A. (2019). Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access*, Vol. 7, pp. 82848–82866. https://doi.org/10.1109/ACCESS.2019.2924343

S26 Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

S27 Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

S28 Dolgopolovas, V., Dagienė, V., Jasutė, E., & Jevsikova, T. (2019). Design science research for computational thinking in constructionist education: A pragmatist perspective. *Problemos*, Vol. 95, pp. 144–159. https://doi.org/10.15388/Problemos.95.12

S29 Durak, H. Y., & 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. https://doi.org/10.1016/j.compedu.2017.09.004

S30 Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, *10*(2), 173–197. https://doi.org/10.30935/cet.554493

S31 Fletcher, G. H. L., & Lu, J. J. (2009). Education: Human computing skills: Rethinking the K-12 experience. *Communications of the ACM*, *52*(2), 23–25. https://doi.org/10.1145/1461928.1461938

S32 Fronza, I., El Ioini, N., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education*, Vol. 17. https://doi.org/10.1145/3055258

S33 Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cárdenas, L., Pantano, P., & Bilotta, E. (2019). Lesson planning by computational thinking skills in Italian pre-service teachers. *Informatics in Education*, Vol. 18, pp. 69–104. https://doi.org/10.15388/infedu.2019.04

S34 García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, *80*, 407–411. https://doi.org/10.1016/j.chb.2017.12.005

S35 Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: exploring computational thinking through code analysis. *Interactive Learning Environments*, Vol. 26, pp. 386–401. https://doi.org/10.1080/10494820.2017.1337036

S36 Garneli, V., & Chorianopoulos, K. (2019). The effects of video game making within science content on student computational thinking skills and performance. *Interactive Technology and Smart Education*. https://doi.org/10.1108/ITSE-11-2018-0097

S37 Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Transactions on Computing Education*, Vol. 17. https://doi.org/10.1145/3105910

S38 Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, Vol. 42, pp. 38–43. https://doi.org/10.3102/0013189X12463051

S39 Grover, S., 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. https://doi.org/10.1080/08993408.2015.1033142

S40 Günbatar, M. S. (2019). Computational thinking within the context of professional life: Change in CT skill from the viewpoint of teachers. *Education and Information Technologies*, Vol. 24, pp. 2629–2652. https://doi.org/10.1007/s10639-019-09919-x

S41 Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, Vol. 17, pp. 229–244. https://doi.org/10.15388/infedu.2018.12

S42 Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, Vol. 27, pp. 628–644. https://doi.org/10.1080/10494820.2019.1610451

S43 Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, Vol. 126, pp. 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

S44 Ioannidou, A., Bennett, V., Repenning, A., Koh, H., & Basawapatna, A. (2011). Computational Thinking Patterns Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom. *Annual Meeting of the American Educational Research Association (AERA)*, *2*. Retrieved from http://www.agentsheets.com

S45 Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, Vol. 82, pp. 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

S46 Israel-Fishelson, R., & Hershkovitz, A. (2019). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research*. https://doi.org/10.1177/0735633119887187

S47 Jeong, Y.-S., & Sung, Y.-H. (2019). The effect of network-based PUMA teaching-learning model on information literacy, computational thinking, and communication skills. *Universal Journal of Educational Research*, Vol. 7, pp. 103–113. https://doi.org/10.13189/ujer.2019.071512

S48 Jun, S., Han, S., & Kim, S. (2017). Effect of design-based learning on improving computational thinking. *Behaviour and Information Technology*, Vol. 36, pp. 43–53. https://doi.org/10.1080/0144929X.2016.1188415

S49 Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, Vol. 59, pp. 26–27. https://doi.org/10.1145/2955114

S50 Kale, U., Akcaoglu, M., Cullen, T., & Goh, D. (2018). Contextual Factors Influencing Access to Teaching Computational Thinking. *Computers in the Schools*, Vol. 35, pp. 69–87. https://doi.org/10.1080/07380569.2018.1462630

S51 Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational What? Relating Computational Thinking to Teaching. *TechTrends*, Vol. 62, pp. 574–584. https://doi.org/10.1007/s11528-018-0290-9

S52 Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal Of Modern Computing*, *4*(3), 583–596.

S53 Kim, Y.-M., & Kim, J.-H. (2016). Application of a software education program developed to improve computational thinking in elementary school girls. *Indian Journal of Science and Technology*, Vol. 9. https://doi.org/10.17485/ijst/2016/v9i44/105102

S54 Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. *Proceedings - 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2010*, (December), 59–66. https://doi.org/10.1109/VLHCC.2010.17

S55 Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers and Education*, Vol. 127, pp. 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

S56 Korkmaz, Ö., & Bai, X. (2019). Adapting computational thinking scale (CTS) for chinese high school students and their thinking scale skills level. *Participatory Educational Research*, *6*(1), 10–26. https://doi.org/10.17275/per.19.2.6.1

S57 Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

S58 Kukul, V., & Karataş, S. (2019). Computational thinking self-efficacy scale: Development, validity and reliability. *Informatics in Education*, Vol. 18, pp. 151–164. https://doi.org/10.15388/infedu.2019.07

S59 Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education*, Vol. 17, pp. 321–340. https://doi.org/10.15388/infedu.2018.17

S60 Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing Teachers to Engage Rural Students in Computational Thinking Through Robotics, Game Design, and Culturally Responsive Teaching. *Journal of Teacher Education*, Vol. 69, pp. 386–407. https://doi.org/10.1177/0022487117732317

S61 Ling, U. L., Saibin, T. C., Labadin, J., & Aziz, N. A. (2018.). Preliminary Investigation: Teachers' Perception on Computational Thinking Concepts. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(2-9), 23-29.

S62 Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, Vol. 41, pp. 51–61. https://doi.org/10.1016/j.chb.2014.09.012

S63 Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, Vol. 80, pp. 470–477. https://doi.org/10.1016/j.chb.2017.09.025

S64 Mishra, P., Cain, W., Sawaya, S., & Henriksen, D. (2013). Rethinking Technology & Creativity in the 21st Century: A Room of Their Own. *TechTrends*, *57*(4), 5–9. https://doi.org/10.1007/s11528-013-0668-7

S65 Monteiro, I. T., Salgado, L. C. de C., Mota, M. P., Sampaio, A. L., & de Souza, C. S. (2017). Signifying software engineering to computational thinking learners with AgentSheets and PoliFacets. *Journal of Visual Languages and Computing*, *40*, 91–112. https://doi.org/10.1016/j.jvlc.2017.01.005

S66 Moreno León, J., Robles, G., & Román González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia*, (46), 6.

S67 Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, Vol. 33, pp. 61–76. https://doi.org/10.14742/ajet.3521

S68 Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*. https://doi.org/10.1080/20004508.2019.1627844

S69 Ozturk, Z., Dooley, C. M. M., & Welch, M. (2018). Finding the Hook: Computer Science Education in Elementary Contexts. *Journal of Research on Technology in Education*, *50*(2), 149–163. https://doi.org/10.1080/15391523.2018.1431573

S70 Pinkard, N., Martin, C. K., & Erete, S. (2019). Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments*, *0*(0), 1–15. https://doi.org/10.1080/10494820.2019.1636070

S71 Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Emerging Research, Practice, and Policy on Computational Thinking. *Emerging Research, Practice, and Policy on Computational Thinking*, 291–305. https://doi.org/10.1007/978-3-319-52691-1

S72 Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., … Gluck, F. (2015). Scalable Game Design. *ACM Transactions on Computing Education*, *15*(2), 1–31. https://doi.org/10.1145/2700517

S73 Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 265–269. https://doi.org/10.1145/1734263.1734357

S74 Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

S75 Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2019). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1612448

S76 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*, Vol. 72, pp. 678–691. https://doi.org/10.1016/j.chb.2016.08.047

S77 Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior*, *80*, 441–459. https://doi.org/10.1016/j.chb.2017.09.030

S78 Román-González, M., Pérez-González, J., & Moreno-León, J. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*, *18*, 47–58. https://doi.org/10.1016/j.ijcci.2018.06.004

S79 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 and Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

S80 Selby, C. (2013). Computational Thinking: The Developing Definition. *ITiCSE Conference 2013*, 5–8.

S81 Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, Vol. 18, pp. 351–380. https://doi.org/10.1007/s10639-012-9240-x

S82 Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469–495. https://doi.org/10.1007/s10639-016-9482-0

S83 Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, Vol. 22, pp. 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

S84 Snow, E., Rutstein, D., Basu, S., Bienkowski, M., & Everson, H. T. (2019). Leveraging Evidence-Centered Design to Develop Assessments of Computational Thinking Practices. *International Journal of Testing*, Vol. 19, pp. 103–127. https://doi.org/10.1080/15305058.2018.1543311

S85 Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, *28*(2), 347–376. https://doi.org/10.1007/s10798-017-9400-9

S86 von Wangenheim, C. G., Hauck, J. C. R., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., &

Azevedo, L. F. (2018). CodeMaster - Automatic assessment and grading of app inventor and snap! Programs. *Informatics in Education*, *17*(1), 117–150. https://doi.org/10.15388/infedu.2018.08

S87 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*, Vol. 20, pp. 715–728. https://doi.org/10.1007/s10639-015-9412-6

S88 Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, Vol. 25, pp. 127–147. https://doi.org/10.1007/s10956-015-9581-5

S89 Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning*, Vol. 6, pp. 1–17. https://doi.org/10.4018/IJGBL.2016010101

S90 Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). Werner, Linda Denner, Jill Campe, Shannon Kawamoto, Damon Chizuru. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.3758/BF03196322

S91 Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

S92 Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

S93 Witherspoon, E. B., & Schunn, C. D. (2019). Teachers' goals predict computational thinking gains in robotics. *Information and Learning Science*, Vol. 120, pp. 308–326. https://doi.org/10.1108/ILS-05-2018-0035

S94 Xing, W. (2019). Large-scale path modeling of remixing to computational thinking. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1573199

S95 Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, Vol. 14. https://doi.org/10.1145/2576872

S96 Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, Vol. 60, pp. 55–62. https://doi.org/10.1145/2994591

S97 Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, Vol. 24, pp. 929–951. https://doi.org/10.1007/s10639-018-9801-8

S98 Yasar, O. (2018). Viewpoint a new perspective on computational thinking: Addressing its cognitive essence, universal value, and curricular practices. *Communications of the ACM*, *61*(7), 33–39. https://doi.org/10.1145/3214354

S99 Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, 141(June), 103607. https://doi.org/10.1016/j.compedu.2019.103607

S100 Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education*, *141*(July), 103633. https://doi.org/10.1016/j.compedu.2019.103633

S101 Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. In *Journal of Educational Computing Research* (Vol. 53). https://doi.org/10.1177/0735633115608444

Appendix B. Supplementary material

Supplementary material presents the recording of the elements of CT Areas in the selected studies. More specifically the following are included:

1. Search Results (tab 2-7)
2. List of the selected studies.
3. Contribution of studies: Which studies contribute to each CT Area.
4. Knowledge Base Area: Which studies include each CT element.
5. Learning Strategies Area: Which studies include each learning strategy.
6. Assessment Area: Which studies include each assessment method.
7. Factors Area: Which studies include each factor.
8. Tools Area: Which studies include each tool.
9. Capacity Building Area: Which studies discuss each topic of Capacity Building Area.

References

Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, Vol. 23, pp. 1501–1514. https://doi.org/10.1007/s10639-017-9675-1

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, Vol. 55, pp. 832–835. https://doi.org/10.1093/comjnl/bxs074

Alfayez, A. A., & Lambert, J. (2019). Exploring Saudi Computer Science Teachers' Conceptual Mastery Level of Computational Thinking Skills. *Computers in the Schools*, Vol. 36, pp. 143–166. https://doi.org/10.1080/07380569.2019.1639593

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30–55. https://doi.org/10.1016/j.ijcci.2018.10.004

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*. https://doi.org/10.1016/j.chb.2019.106185

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology and Society*, Vol. 19, pp. 47–57. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85000838214&partnerID=40&md5=3f014c90dafb945e90c9552f5a6ef17f

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, Vol. 75, pp. 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *Proceedings - Frontiers in Education Conference, FIE*. https://doi.org/10.1109/FIE.2012.6462365

Barr, V., & Stephenson, C. (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), 48–54. https://doi.org/10.1145/1929887.1929905

Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior*, *80*, 412–419. https://doi.org/10.1016/j.chb.2017.04.058

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*, Vol. 27, pp. 5–53. https://doi.org/10.1007/s11257-017-9187-0

Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, *42*(3), 53–72. https://doi.org/10.14221/ajte.2017v42n3.4

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*, 1–25. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Buitrago Flórez, F., 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*, Vol. 87, pp. 834–860. https://doi.org/10.3102/0034654317710096

Carlborg, N., Tyrén, M., Heath, C., & Eriksson, E. (2019). The scope of autonomy when teaching computational thinking in primary school. *International Journal of Child-Computer Interaction*, *21*, 130–139. https://doi.org/10.1016/j.ijcci.2019.06.005

Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93–100. https://doi.org/10.1016/j.ijcci.2018.06.005

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, Vol. 109,

pp. 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, Vol. 62, pp. 563–573. https://doi.org/10.1007/s11528-018-0292-7

Clark, D. B., & Sengupta, P. (2019). Reconceptualizing games for integrating computational thinking and science as practice: collaborative agent-based disciplinarily-integrated games. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1636071

Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). Education: A future for computing education research. *Communications of the ACM*, *57*(11), 34–36. https://doi.org/10.1145/2668899

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. Retrieved from Computing at Schools. website: https://community.computingatschool.org.uk/resources/2324/single

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education*, Vol. 18, pp. 41–67. https://doi.org/10.15388/infedu.2019.03

CSTA & ISTE. (2011). Operational definition of computational thinking. Retrieved from https://www.iste.org/explore/Solutions/Computational-thinking-for-all

Da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, Vol. 18, pp. 17–39. https://doi.org/10.15388/infedu.2019.02

De Souza, A. A., Barcelos, T. S., Munoz, R., Villarroel, R., & Silva, L. A. (2019). Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access*, Vol. 7, pp. 82848–82866. https://doi.org/10.1109/ACCESS.2019.2924343

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Dolgopolovas, V., Dagienė, V., Jasutė, E., & Jevsikova, T. (2019). Design science research for computational thinking in constructionist education: A pragmatist perspective . *Problemos*, Vol. 95, pp. 144–159. https://doi.org/10.15388/Problemos.95.12

Durak, H. Y., & 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. https://doi.org/10.1016/j.compedu.2017.09.004

Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, *10*(2), 173–197. https://doi.org/10.30935/cet.554493

Fletcher, G. H. L., & Lu, J. J. (2009). Education: Human computing skills: Rethinking the K-12 experience. *Communications of the ACM*, *52*(2), 23–25. https://doi.org/10.1145/1461928.1461938

Fronza, I., El Ioini, N., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. ACM Transactions on Computing Education, Vol. 17. https://doi.org/10.1145/3055258

Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cárdenas, L., Pantano, P., & Bilotta, E. (2019). Lesson planning by computational thinking skills in Italian pre-service teachers. *Informatics in Education*, Vol. 18, pp. 69–104. https://doi.org/10.15388/infedu.2019.04

García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university

education. *Computers in Human Behavior*, *80*, 407–411. https://doi.org/10.1016/j.chb.2017.12.005

Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: exploring computational thinking through code analysis. *Interactive Learning Environments*, Vol. 26, pp. 386–401. https://doi.org/10.1080/10494820.2017.1337036

Garneli, V., & Chorianopoulos, K. (2019). The effects of video game making within science content on student computational thinking skills and performance. *Interactive Technology and Smart Education*. https://doi.org/10.1108/ITSE-11-2018-0097

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering, 9(4), 248–260. https://doi.org/10.1007/s00766-004-0204-6

Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Transactions on Computing Education*, Vol. 17. https://doi.org/10.1145/3105910

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, Vol. 42, pp. 38–43. https://doi.org/10.3102/0013189X12463051

Grover, S., 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. https://doi.org/10.1080/08993408.2015.1033142

Günbatar, M. S. (2019). Computational thinking within the context of professional life: Change in CT skill from the viewpoint of teachers. *Education and Information Technologies*, Vol. 24, pp. 2629–2652. https://doi.org/10.1007/s10639-019-09919-x

Heintz, F., Mannila, L., & Farnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *Proceedings - Frontiers in Education Conference, FIE*, *2016-Novem*. https://doi.org/10.1109/FIE.2016.7757410

Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, Vol. 27, pp. 628–644. https://doi.org/10.1080/10494820.2019.1610451

Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, Vol. 17, pp. 229–244. https://doi.org/10.15388/infedu.2018.12

Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, Vol. 126, pp. 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Hsu, Y.-C., Irie, N. R., & Ching, Y.-H. (2019). Computational Thinking Educational Policy Initiatives (CTEPI) Across the Globe. *TechTrends*. https://doi.org/10.1007/s11528-019-00384-4

Ioannidou, A., Bennett, V., Repenning, A., Koh, H., & Basawapatna, A. (2011). Computational Thinking Patterns Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom. *Annual Meeting of the American Educational Research Association (AERA)*, *2*. Retrieved from http://www.agentsheets.com

Israel-Fishelson, R., & Hershkovitz, A. (2019). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research*. https://doi.org/10.1177/0735633119887187

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, Vol. 82, pp. 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, Vol. 59, pp. 26–27. https://doi.org/10.1145/2955114

Kale, U., Akcaoglu, M., Cullen, T., & Goh, D. (2018). Contextual Factors Influencing Access to Teaching Computational Thinking. *Computers in the Schools*, Vol. 35, pp. 69–87. https://doi.org/10.1080/07380569.2018.1462630

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational What? Relating Computational Thinking to Teaching. *TechTrends*, Vol. 62, pp. 574–584. https://doi.org/10.1007/s11528-018-0290-9

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal Of Modern Computing*, 4(3), 583–596.

Kim, Y.-M., & Kim, J.-H. (2016). Application of a software education program developed to improve computational thinking in elementary school girls. *Indian Journal of Science and Technology*, Vol. 9. https://doi.org/10.17485/ijst/2016/v9i44/105102

Koehler, M. J., & Mishra, P. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge PUNYA MISHRA. *Teachers College Record*, *108*(6), 1017–1054. Retrieved from http://one2oneheights.pbworks.com/f/MISHRA_PUNYA.pdf

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers and Education*, Vol. 127, pp. 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

Korkmaz, Ö., & Bai, X. (2019). Adapting computational thinking scale (CTS) for chinese high school students and their thinking scale skills level. *Participatory Educational Research*, *6*(1), 10–26. https://doi.org/10.17275/per.19.2.6.1

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Kukul, V., & Karataş, S. (2019). Computational thinking self-efficacy scale: Development, validity and reliability. *Informatics in Education*, Vol. 18, pp. 151–164. https://doi.org/10.15388/infedu.2019.07

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education*, Vol. 17, pp. 321–340. https://doi.org/10.15388/infedu.2018.17

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing Teachers to Engage Rural Students in Computational Thinking Through Robotics, Game Design, and Culturally Responsive Teaching. *Journal of Teacher Education*, Vol. 69, pp. 386–407. https://doi.org/10.1177/0022487117732317

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, Vol. 41, pp. 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, Vol. 80, pp. 470–477. https://doi.org/10.1016/j.chb.2017.09.025

Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *BMJ (Online)*, *339*(7716), 332–336. https://doi.org/10.1136/bmj.b2535

Moreno León, J., Robles, G., & Román González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia*, (46), 6.

Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, Vol. 33, pp. 61–76. https://doi.org/10.14742/ajet.3521

Mylopoulos, J. (1992). Conceptual modelling and Telos, in: P. Loucopoulos, R. Zicari. *Conceptual Modeling,*

Databases, and Case An Integrated View of Information Systems Development. Wiley New York, 1992, pp. 49–68

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*. https://doi.org/10.1080/20004508.2019.1627844

Ozturk, Z., Dooley, C. M. M., & Welch, M. (2018). Finding the Hook: Computer Science Education in Elementary Contexts. *Journal of Research on Technology in Education*, *50*(2), 149–163. https://doi.org/10.1080/15391523.2018.1431573

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

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, *22*(2), 421–443. https://doi.org/10.1007/s10639-016-9475-z

Piaget, J. (1970). *Genetic Epistemology*. New York: Columbia University Press.

Pinkard, N., Martin, C. K., & Erete, S. (2019). Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments*, *0*(0), 1–15. https://doi.org/10.1080/10494820.2019.1636070

Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Emerging Research, Practice, and Policy on Computational Thinking. *Emerging Research, Practice, and Policy on Computational Thinking*, 291–305. https://doi.org/10.1007/978-3-319-52691-1

Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., … Gluck, F. (2015). Scalable Game Design. *ACM Transactions on Computing Education*, *15*(2), 1–31. https://doi.org/10.1145/2700517

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

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*, Vol. 72, pp. 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior*, *80*, 441–459. https://doi.org/10.1016/j.chb.2017.09.030

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 and Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Selby, C. (2013). Computational Thinking : The Developing Definition. *ITiCSE Conference 2013*, 5–8.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, Vol. 18, pp. 351–380. https://doi.org/10.1007/s10639-012-9240-x

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher’s perspective. *Education and Information Technologies*, *22*(2), 469–495. https://doi.org/10.1007/s10639-016-9482-0

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, Vol. 22, pp. 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Siau, K., & Tan, X. (2005). Improving the quality of conceptual modeling using cognitive mapping techniques. *Data and Knowledge Engineering*, *55*(3), 343–365. https://doi.org/10.1016/j.datak.2004.12.006

Snow, E., Rutstein, D., Basu, S., Bienkowski, M., & Everson, H. T. (2019). Leveraging Evidence-Centered Design to Develop Assessments of Computational Thinking Practices. *International Journal of Testing*, Vol. 19, pp. 103–127. https://doi.org/10.1080/15305058.2018.1543311

von Wangenheim, C. G., Hauck, J. C. R., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F.

(2018). CodeMaster - Automatic assessment and grading of app inventor and snap! Programs. *Informatics in Education*, *17*(1), 117–150. https://doi.org/10.15388/infedu.2018.08

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*, Vol. 20, pp. 715–728. https://doi.org/10.1007/s10639-015-9412-6

Vygotsky, L. S. (1978). *Mind inSociety: TheDevelopment ofHigher Psychological Processes.* Harvard University Press.

Wand, Y., & Weber, R. (2002). Research commentary: Information systems and conceptual modeling - A research agenda. *Information Systems Research*. https://doi.org/10.1287/isre.13.4.363.69

Webster, J., & Watson, R. (2002). Analyzing the past to prepare for the future: writing a literature review. *MIS Quarterly*, *26(2)*, 13–23.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, Vol. 25, pp. 127–147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning*, Vol. 6, pp. 1–17. https://doi.org/10.4018/IJGBL.2016010101

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *SIGCSE'12 - Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. https://doi.org/10.1145/2157136.2157200

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, Vol. 49, pp. 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 366, pp. 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Witherspoon, E. B., & Schunn, C. D. (2019). Teachers' goals predict computational thinking gainsin robotics. *Information and Learning Science*, *120*(5–6), 308–326. https://doi.org/10.1108/ILS-05-2018-0035

Xing, W. (2019). Large-scale path modeling of remixing to computational thinking. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1573199

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, Vol. 14. https://doi.org/10.1145/2576872

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, Vol. 60, pp. 55–62. https://doi.org/10.1145/2994591

Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, *24*(1), 929–951. https://doi.org/10.1007/s10639-018-9801-8

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, Vol. 141. https://doi.org/10.1016/j.compedu.2019.103607

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education*, *141*(July), 103633. https://doi.org/10.1016/j.compedu.2019.103633

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. In *Journal of Educational Computing Research* (Vol. 53). https://doi.org/10.1177/0735633115608444