Incorporating Computational Thinking into Education: From Teacher Training to Student Mastery

Antonella Nuzzaci

University of Messina, Department of Cognitive Sciences, Psychology, Education, and Cultural Studies, Italy

Received: June 3, 2024 Accepted: July 5, 2024 Published: July 6, 2024

doi:10.5296/jet.v11i2.21942 URL: https://doi.org/10.5296/jet.v11i2.21942

Abstract

The concept of computational thinking originated in the 1950s and 1960s as “algorithmic thinking” (Denning, 2009), which involves using a systematic and precise sequence of steps to solve problems, potentially using computers to automate the process. Today, 'computational thinking' is defined as is a solving problem process that involves formulating problems and solutions in a way that can be represented and solved through a series of computational steps, designing systems and understanding human behaviour, based on fundamental computer concepts. It includes processes such as abstraction and decomposition when dealing with complex tasks or designing extended systems (Wing, 2006). From an educational point of view, the challenge is to identify the cognitive skills that are expected of a person with this type of thinking and their ability to apply them in practice. Computational thinking encompasses a set of thought processes that originate from computer science, but are applicable in various fields. However, it is often wrongly perceived as “technological thinking”, which implies a mindset aimed exclusively at the effective use of technologies. The contribution points out that the lack of a clear definition hinders the integration of this concept into teacher training and broader educational contexts. If understood correctly, computational thinking could significantly improve understanding of the procedural aspects of knowledge, which are very useful for teaching. In this sense, the contribution also describes an exploratory research on the opinions of primary school teachers in central Italy aimed at understanding the perceived benefits that the incorporation of computational thinking could bring in a teaching context.

Keywords: computational thinking, computational thinking education, educational technology, teacher training

1. Introduction

In the era of participatory culture, one of the critical media literacy skills for teachers and students in the 21st century is the ability to locate, analyse, generalise and visualise information gathered and acquired from multiple media sources (Jenkins et al., 2010). The constant influx of multimodal data requires students to use not only their traditional literacy skills, but also new skills, such as digital and technological skills, to expand, integrate and strengthen their interpretive knowledge repertoires. Alongside this cognitive set, and not superimposed on it, there is another family of higher-order skills related to computational thinking, some of whose characteristics are addressed in this paper in relation to the function they perform within the teaching-learning processes.

The concept of computational thinking first appeared in the 1950s and 1960s under the term "algorithmic thinking" (Denning, 2009). It is immediately associated with the use and application of a structured and precise sequence of steps to deal with problems, and, when necessary, with the use of a computer to automate these processes. Today, the term "computational thinking" is generally associated with the skills and methods for solving problems (Jonassen, 2008; Kahney, 1993), designing systems and understanding human behaviour, drawing on the fundamental concepts of computer science, a distinct and recognised discipline (Denning, 2013; Denning & Tedre, 2019; 2021) that deals with algorithmic problem solving. Computational thinking (Csizmadia et al., 2015) involves the use of abstraction and decomposition skills when an individual approaches a complex task or even the design of a large system (Wing, 2006; 2008). In the book Computational Thinking, Peter J. Denning and Matti Tedre (2019) define computational thinking as a set of mental and practical skills aimed at designing computations that lead people to use computers to automate certain tasks and to explain and interpret the world as a complex of information processes aimed at developing some indispensable skills. Selby and Wollard (2013) try to construct a definition of computational thinking, starting from the justification of the inclusion or exclusion of the main prospective terms, while eliminating the less defined elements, as shown in the table below.

Table 1. Computational Thinking Definition Terminology (Selly & Wollard, 2013)

Terms	**Status**	**Justification**
A thought process	Includes	Consensus found in the literature
Abstraction	Includes	Consensus found in the literature
Decomposition	Includes	Consensus found in the literature
Logical thinking	Exclude	Broad term, not-well defined
Algorithmic thinking	Includes	Well-defined across multiple disciplines
Problem solving	Exclude	Broad term, evidences the use of skills; develops acquisition of skills
Evaluation	Includes	Well-defined across multiple disciplines
Generalization	Includes	Well-defined concept, although the term may not be familiar
Systems design	Exclude	Evidences the use of skills
Automation	Exclude	Evidences the use of skills
Computer science content	Exclude	Evidences the use of skills
Modeling, simulation, and	Exclude	Evidences the use of skills in their creation; manipulation develops acquisition of skills

Selby and Woollard (2013), building on Wing's statement, describe a five-dimensional model of computational thinking that reflects the ability to:

- Abstraction, to understand problems by filtering information and reducing unnecessary detail;
- Decomposition, to break the problem down into smaller, solvable problems.
- Algorithmic thinking, to find the solution to the problem step by step;
- Evaluation, to assess the effectiveness of the solution;
- Generalisation, to be able to generalise the solution to a wider range of problems.

It is understood as the ability to interpret the world through algorithmically controlled input-to-output transformations (Denning, 2009), involving thought processes related to the formulation of problems and their solutions, in such a way that they are represented in a form that can be effectively executed by an information processing agent. Specifically, therefore, it is interpreted as a set of mental processes that are used to model a situation and specify the ways in which an information processing agent can effectively operate within the situation itself to achieve one or more externally provided goals (Nardelli, 2019). In other words, such a construct would involve learning to think, represent, and solve problems that require a combination of human cognitive and computational abilities.

However, one of the central issues is the relationship between the process of processing information and the computation required to understand and then control that processing (Denning, 2009; Furber, 2012). Another issue would be the connection between computational thinking and formal operations (Inhelder & Piaget, 1955), where the components of the former would be linked to the latter.

It should also be remembered that researchers often associate computational thinking mainly with content and procedural knowledge rather than with formal reasoning skills. This opens up the possibility of developing an idea of computational thinking even in the presence of weaknesses related to formal reasoning, and leads us to reflect on an aspect that seems to be central to the current debate. However, the nature of their relationship and their specific interaction within teaching-learning processes is still unclear.

From an educational perspective, the question is how a person is expected to develop specific cognitive skills associated with this type of thinking and their ability to use these skills effectively. If computational thinking is presented as a set of problem-solving processes rooted in computing but applicable across domains, it should not be misinterpreted as 'technological thinking', which implies a mindset developed only through the appropriate and relevant use of technology in context.

In this sense, the present paper focuses precisely on analysing the origin and the persistence of the overlap between technological and computational thinking, which still seems to be very present in the imagination and in the common sense, but which, in fact, now represents a kind of "conceptual misrepresentation" that seems quite unacceptable at the moment.

Indeed, the review of the literature (Espinal, Vieira, & Magana, 2024) on the conceptualisation of computational thinking suggests that there is no agreement on a clear definition of it, which sometimes reveals even scarce conceptual convergences, since its reformulations over time have been frequent and common efforts to characterise it have been difficult.

It is important to recognise here that the lack of a clear definition of 'computational thinking' makes it difficult both to develop clear curricular pathways at all levels of education and to use it appropriately in the contexts of initial and in-service teacher training, Clarifying this could be crucial in taxonomically defining the knowledge and skills of teachers and students engaged in this kind of thinking.

2. Computational Thinking and Its Relationships

In many cases, computational thinking is also conceived as a process that is not necessarily achieved by learning computer programming or computer science per se (Lu & Fletcher, 2009; Lye & Koh, 2014; Wing, 2008; Voogt, et al., 2015), but that can be implemented through teaching (Corradini, Lodi, & Nardelli. 2018). In all cases, computational thinking involves a set of skills that people develop over time to solve problems. Although not all studies directly mention the term 'computational thinking', the evidence literature has shown over time how its key features and components actually encourage students to remain reality-centred. This type of thinking appears to underpin the learning of many areas of knowledge, such as science, for example, identifying relationships between variables, predicting behaviours and their possible consequences, or even constructing models based on large sets of data, using mathematical concepts to support explanations and arguments. In simpler terms, it can be argued that if someone is able to solve problems and provide evidence in support of science, they are likely to have a greater understanding of both the substantive ideas underlying science itself and its procedural aspects.

Baird and Borich (1987) suggest that formal reasoning and scientific process skills may be two traits afferent to the same cognitive structure. Demonstrating competence in integrated scientific process skills is said to require the use of higher-order thinking skills, because competence in scientific process skills involves the ability to apply learned material to new and concrete situations, to analyse the relationships between parties and identify the organisational principles involved, to synthesise parts to form a new whole, and to evaluate or judge the value of material, such as judging the appropriateness of conclusions supported by data (Baird & Borich, 1987). In their paper, Repenning, Basawapatna and Escherle (2017) examine the foundations of computational thinking tools, defining it as a set of cognitive skills and methods required to solve complex problems through computational processes. Such thinking includes concepts such as abstraction, decomposition, pattern recognition and algorithm design. The authors describe various educational tools designed to teach computational thinking, such as visual programming environments and interactive simulations. They identify several design principles that make computational thinking tools effective. These principles include ease of use, the ability to stimulate creativity, immediate feedback and the ability to support different learning styles. In particular, they present case studies and empirical evidence demonstrating the effectiveness of computational thinking tools in the classroom and report on successful examples where students have improved their problem-solving skills.

Problem solving in science is precisely represented by the involvement of two types of understanding, substantive and procedural. When scientists solve problems, they use both the former and the latter (Roberts & Gott, 1999).

Some researchers go so far as to point out the link between formal reasoning and procedural knowledge in science (Chandron, Treagust & Tobin, 1987) and to identify operations of formal reasoning as essential skills for learning science (Bitner, 1991).

The ability to carry out operations such as conservation, variable identification, probabilities, proportions, combinations and correlations is recognised as crucial for understanding scientific concepts.

However, the relationship between computational thinking and scientific learning remains poorly understood. Some researchers see computational thinking as a distinct domain (Chambers, 1988; Swartz & Perkins, 1990), but its definitions do not seem to be fully integrated into science learning environments.

However, it should also be noted that some skills, such as distinguishing a hypothesis from a problem or detecting errors in arguments, are common elements in both the definitions of computational thinking and scientific thinking. Equally common are the dispositions of openness and flexibility (Facione, 2007; 2011; d'Angelo, 1971), as well as the concern for procedural and methodological precision, among others. In this context, it is also worth highlighting, without claiming to be exhaustive, the importance of these relationships, which underpin formal reasoning, for the development of dispositions and attitudes that can influence learning outcomes at the cognitive level.

3. Computational Thinking and Problem-Solving

Papert (1980, p. 183) coined the term "computational thinking" to describe the practice of procedural thinking taught to children, arguing that learning is most effective when students "construct knowledge" and develop it by doing rather than by telling. In fact, within his constructivist approach to learning, he supported the importance of some elements such as self-directed learning, design learning, meaningful representations, facilitation-based education and the use of technology in context to support the learning processes in school, which led to the central idea of "Mindstorms" (Papert, 1980) or the transition from "learning to code" to "programming to learn", which has been a difficult acquisition to introduce in the field of teacher training and education in general. Programming is based on a process of rationalisation of design and its application and the regulation of a practical activity according to scientific criteria. However, if it is true that computer science is not just programming, we must remember that in the educational field it allows the development of software and the creation of computational artefacts such as visualisations and the development and execution of algorithms, while offering opportunities to create new knowledge and solve problems. The idea then is to go beyond learning a particular type of programming language and provide a learning experience that can enables students and teachers to use computational thinking to monitor other processes and to act by modifying the situation in which they find themselves.

The belief that computational thinking is a metacognitive or higher-order skill that can be learned through programming, and that teachers and students who acquire it in a particular domain can then become better problem solvers in that domain or in other domains (transfer), has been reinforced over time, and with it the idea that computing should occupy a central place in educational processes and in the initial and continuing training of teachers. However, the proponents of these theses have, for various reasons, emphasised the role that the relationship between programming and computational thinking plays at the procedural level. In this sense, the teacher becomes a learning technologist, not because he uses technologies, but because he is a producer and creator of intervention schemes in different teaching situations on the basis of learned scientific criteria. Programming, using computational thinking in both learning and teaching, would help to improve general thinking skills, such as logical thinking, which "sharpens the mind". Critics of this approach have pointed out its critical aspects and questioned the hypothesis of transferability of this type of skills to other domains.

In any case, programming involves a complex network of skills, including mathematical, conditional, analogical, procedural, temporal and memory skills, although the role that these skills play in teaching-learning processes and the extent to which they are interrelated and transferable is not always clear. This has led to a heated debate about the introduction of computing education in schools.

It should also be remembered that several reasons have been put forward to explain why the construction of computational thinking should aim to develop procedural understanding at school and thus strengthen the cultural profile of teachers and students. The first concerns the aims of education. It is, of course, essential to clarify how education prepares students to become active citizens in an ever-changing society and to deal with real-life problems. This explains the importance of developing procedural understanding (Roberts & Gott, 1999) so that all individuals can make appropriate decisions in context. However, when we talk about 'problem', it is important to point out that this term is also broad and full of pitfalls, since it refers not only to those mathematically well-defined problems, whose solutions are fully analysable, such as a proof, an algorithm or a program, but also to all those problems that concern the real world and everyday life, whose solutions may exist in the form of large, complex systems.

A problem in learning is a situation or condition that requires a solution and can be addressed through critical thinking, creativity and the application of prior knowledge and skills. The problem solving process is fundamental to cognitive development and includes the stages of understanding the nature and scope of the problem, analysing its components and the variables involved in the problem, generating possible solution strategies to solve the problem, evaluating the pros and cons of the proposed solution, applying and implementing the chosen solution, evaluating the effectiveness of the solution and reflecting on the whole process. In this sense, in the classroom, a teacher who is able to present well-defined and relevant problems can facilitate students' active learning, engagement and the development of critical thinking skills, which has a profound impact on teaching practices and educational methodologies.

Wing (2006; 2008) used the word 'problem' simply to refer to something that needs to be solved. Since solving a problem is only one instance of a situation in which a specified goal is to be achieved, computational thinking sees the thought process involved in modelling a situation and specifying the ways in which an information-processing agent can operate effectively within it to achieve one or more specified goals. This assumption sees the mental processes involved in formulating problems and their solutions in such a way that they are represented in a form that can be effectively executed by an information-processing "agent". The mental processes involved in formulating problems and their solutions can therefore be represented as computational and algorithmic steps.

However, the fundamental issues remain: to fully demonstrate whether the problem-solving process can be generalised and transferred to a wide variety of problems, and to reduce and define 'computational knowledge' so that it is fully operational, teachable and evaluable (Armoni, 2016). This is about the ability to unravel, i.e. to explain in what sense "the way a computer scientist thinks" is different from "the way a biologist thinks". On the other hand, Riley and Hunt (2014, p. 4) had already argued that the best way to characterise computational thinking is to look at how "computer scientists think and reason", presenting the idea of coding as a promoter of the development of creative, motivating and meaningful computational thinking for students, and as such able to make them capable of greater "computational fluidity", i.e. using computational technologies to communicate ideas effectively and creatively (Resnick & Rusk, p. 122). This appears to be a fundamental need in the digital society (Cooper et al., 2016).

It is also worth noting that there is a form of procedural understanding that is necessary to achieve goals of a different nature, including those of a scientific nature. Indeed, Roberts and Gott (2000) argue that forms of procedural knowledge can and should be taught in the same way as any other form of knowledge; for example, one would never expect schoolchildren to understand the importance of the circulatory system or the role of chlorophyll in photosynthesis without explicit teaching, so one cannot expect the same to be true of procedural knowledge.

Computational thinking therefore requires an activity of didactic mediation and explication aimed at enabling students to develop a procedural understanding through both 'theoretical' and 'practical' activities that can be used to teach content and skills. This should facilitate the teacher's choice of modes, activities and strategies (Dalziel, 2010; Dalziel & Dalziel, 2011), which can vary according to students' needs and learning objectives. Tamir et al (1998) have shown that teaching students to think procedurally increases their general understanding of knowledge and involves the use of precise thinking strategies. However, the nature of the cognitive support it provides to teachers and students in practice is still poorly understood empirically.

4. Computer Science, Computational Thinking, and Teaching

Computers have changed the way we think and work, requiring us to adapt our mental processes in order to use and communicate with them more effectively, while at the same time highlighting the importance of information, data and, above all, algorithms, which are increasingly present in everyday life. Many aspects of computational thinking have a long history that predates the invention of computers. Currently, at different levels of education, computational thinking, as shown by Denning and Tedre (2016) when they describe the process of its integration into the school in many countries (as is happening in the Italian field), has generated a broad movement whose evolution has been determined by three important factors that have allowed its development: the transformation of the ways of thinking and practicing computer science within the different disciplinary fields; the affirmation of a relationship between educational research (Frankling, 2015) computer science and the birth of a computational science; the digitalisation of society. By outlining these steps, Denning and Tedre (2019) provide a comprehensive guide for educators and policymakers aiming to integrate computational thinking into educational systems, emphasising the need for a systematic and collaborative approach. They provide an in-depth exploration of the integration of computational thinking into educational curricula and different disciplines, and describe this process of integration as multifaceted, involving several key stages and considerations:

- *Understand computational thinking:* Emphasise the importance of understanding the concept of computational thinking (CT) itself. They define CT as a problem-solving process that involves a set of cognitive skills and techniques that can be applied across domains. These include abstraction, algorithmic reasoning, decomposition, pattern recognition and evaluation.

- *Curriculum development*: The integration process involves designing curricula that incorporate CT principles. This means not only creating stand-alone courses in computing, but also embedding CT concepts in other subject areas. For example, in mathematics students might use algorithmic thinking to solve problems, while in social studies they might use data analysis to understand historical trends.

- *Teacher training*: Effective integration requires educators to be well versed in CT principles and teaching methods. Denning and Tedre discuss the need for professional development programmes that equip teachers with the skills and knowledge necessary to teach CT effectively.

- *Interdisciplinary collaboration*: Integration is most effective when there is collaboration across subject areas. Denning and Tedre highlight the importance of interdisciplinary projects that allow students to apply CT in different contexts, thereby deepening their understanding and skills.

- *Assessment and evaluation*: To ensure that CT integration is successful, there must be robust methods for assessing students' understanding and application of CT principles. This involves developing new forms of assessment that go beyond traditional tests to include project-based assessments and performance tasks.

- *Addressing challenges and barriers*: recognise various challenges in integrating CT, such as resistance to change, lack of resources, and different levels of prior knowledge among students and teachers. They suggest strategies for overcoming these barriers, including building a supportive school culture, securing funding, and creating a climate of trust.

In the 1990s, however, training in computational thinking was mainly associated with university education, while at other levels of education, a handful of courses on the use of computers were promoted in schools, some of which focused on computer literacy and others on a superficial knowledge of programming languages. A critical phase came after 2000, when many people realised the pervasiveness of the computer (Bundy, 2007) in everyday work and private life; and it was only in the following years that teachers and educational policymakers began to agree on the importance of understanding the use of technological knowledge and computer science, the mechanisms of digitisation and the function of digital skills in the 21st century, even if from that moment on, in the common sense, a real synonymy between technologies and information technology began. The concept of algorithm is central to mathematics and computer science, with a long history of pioneering contributions by mathematicians and scientists such as al-Khwarizmi, Ada Lovelace and Alan Turing.

The term “algorithm” which is derived from the name of the Persian mathematician Muhammad ibn Musa al-Khwarizmi (ninth century), whose compendium entitled *book Al-Kitab al-Mukhtasar fi Hisab al-Jabr wal-Muqabala* on calculation by completion and balancing which gave rise to the term “algebra”. Although he did not directly deal with algorithms as we understand them today, his work on systematic methods of solving mathematical problems laid the foundations for the modern concept of the algorithm. It is also thanks to Ada Lovelace (19th century), the first female computer programmer, who wrote an algorithm intended to be executed by a machine, Charles Babbage's analytical engine. His pioneering work is a milestone in the history of algorithms. Until Alan Turing (20th century), who is considered the father of modern computing, who formalised the concept of algorithm with his Turing machine, a theoretical model of computation that made it possible to define what it means for a problem to be solvable by a machine. His famous 1936 paper, *On Computable Numbers, with an Application to the Entscheidungsproblem,* had a huge impact on the development of algorithmic theory.

A classic example of an algorithm is Euclid's algorithm for finding the greatest common divisor (GCD) of two integers:

1. Start with two numbers, *a* and *b*.
2. If *b* is *0*, the MCD is *a*.
3. Otherwise, replace *a* with *b* and *b* with *a%b* (the remainder of the division of *a* by *b*).
4. Repeat steps 2 and 3 until *b* becomes *0*.

This algorithm is known for its efficiency and is one of the oldest known algorithms, dating back to the time of Euclid (around 300 BC).

Over time, however, the concept of algorithms, once obscure, gradually entered people's daily conversations with the advent of online transactions, well-formatted spreadsheets and documents, and the creation of viewable presentations. In other words, these elements became essential for the functioning of the contemporary world.

Procedural knowledge soon ended up being more and more attentive to research, which initiated a conscious, deliberate and systematic reflection on it, gradually revealing, even in formative contexts, that the truth might not be immediately accessible. They were linked to the acquisition and application of procedures for acquiring and communicating knowledge, for looking at different perspectives, for opening windows and visions of the world, and for learning to read reality and data more objectively.

Thus, procedural knowledge today focuses on the development of skills and techniques to seek the truth of the data, emphasising method and form, but not content. Proceduralists are, in fact, practical and pragmatic problem solvers; abstraction and automation are the essence of computational thinking, and an algorithm can be said to be a key procedure for constructing an abstraction. However, there is one aspect of computation that is crucial, and that is the ability to abstract, which for Jean Piaget (1936; 1947) develops through different stages of childhood thinking. His theory of cognitive development describes how children develop an understanding of the world around them and how this understanding changes over time. According to Piaget, the ability to abstract is closely linked to the transition to the stage of formal operations, where adolescents become able to formulate hypotheses and reason deductively. They can consider several possible solutions to a problem and test them systematically (hypothesis-deductive thinking), they can reason on the basis of abstract and logical propositions, regardless of the concrete content. For example, they can understand and manipulate formal algebraic and logical expressions (purposive reasoning), and they can combine different elements in complex ways to explore all possible combinations and relationships between them (combinatorial reasoning).

In this direction, the capacity for abstraction is crucial for intellectual development, as it allows individuals to:

- solve complex and theoretical problems;
- understand advanced scientific and mathematical concepts;
- plan and reflect on the future;
- develop complex philosophical, ethical, and social ideas.

Piaget (1936) pointed out that not everyone achieves the full development of formal operations, and that cognitive development can vary greatly between individuals. However, the ability to abstract represents a major turning point in cognitive thinking and intellectual maturation, as well as the process of generalisation that simplifies complex phenomena or problem-solving procedures (Lee, et al., 2011). A computer system is made up of thousands of subsystems, and to building a subsystem requires abstraction. Abstraction can benefit people by allowing them to use the complex computer system without knowing how the internal system works. In this context, Wing (2006; 2008) spoke of abstraction as the main component of computational thinking and argued that students should be taught abstraction in school as it would allow them to reach higher levels of thinking. He believed that acquiring the ability to abstract can help students develop higher order thinking skills that can accustom them to thinking like real 'computer scientists'.

From here, the literature begins to emphasise the intervention of many factors and the presence of elements attributed to the nature of computational thinking, and scholars begin to try their hand at elaborating real lists, such as the one proposed by Grover and Peg (2013), which identifies some key components that characterise it:

- abstraction and automation;
- systematic information process;
- symbolic systems and representations;
- algorithmic flow control notifications;
- decomposition of the structured problem;
- iterative, recursive, and parallel thinking;
- conditional logic;
- efficiency and performance limits;
- debugging and systematic error detection.

In the light of the most advanced and emerging research constructs in the field of computer science, taking into account the literature, it is important to recognise that the current lack of an exclusive agreed definition of the elements that are an integral to computational thinking makes the challenge of its systematic introduction in the field of education very complex, but extremely interesting, yet extremely interesting. This implies the preliminary clarification of its usefulness in education at all levels and across disciplines (Barr, Harrison, & Conery, 2011), so that curricular paths can be developed for teachers in both initial training and in-service, making them truly computationally competent. It is important to note, however, that significant steps have already been taken in this direction.

In fact, the training of computational thinking is now added to that of reading, writing and arithmetic, which are conceived as families of skills that remain the basis of alphabetic processes. Its relationship with many other dimensions increases its power and use, such as in the case of analytical thinking, which is based on concepts taken from computer science, and in the specific use of heuristics, which consists of an approach to problem solving, learning or discovery that uses practical and sub-optimal methods to produce sufficiently good solutions. Especially when it is not practical to find an optimal or exact solution. In other words, heuristics are strategies or rules of thumb that help you make decisions or solve problems quickly and efficiently.

Representativeness heuristic: Judging the probability of an event based on how much it represents or corresponds to an existing prototype. For example, deciding that a person is a doctor because they look like what we consider the prototype of a doctor.

Availability heuristics: Evaluate the probability of events based on how easily relevant examples come to mind. For example, fearing flying because you are easily reminded of airplane crashes reported in the media.

Recognition heuristic: When comparing two options, if one is recognized and the other is not, you tend to choose the one that is recognized. For example, in a test, choose a familiar answer over an unknown one.

Anchoring heuristics and adjustment: making an estimate starting from an initial value (anchor) and then adjusting it to reach a final answer. For example, estimate the price of a house based on the prices of neighbouring houses (anchor) and then adjust it according to the specific characteristics of the house in question.

Such an approach to problem solving, which involves the application of a general rule or strategy that can lead to the solution of a problem, and a heuristic process that involves finding strategies that generally lead to the correct solution, but do not always guarantee this solution, as when a person asks for directions to an unknown place from a certain place, but could also end up in the wrong place, depending on his understanding and ability to interpret (topological, spatial, territorial, etc.). Therefore, the development of computational thinking is central, from modelling processes and phenomena of reality to reasoning, from formulating and solving problems to comparing and executing procedures and algorithms.

5. Computational Thinking and Teaching-Learning Processes

Incorporating computational thinking into the teaching-learning processes requires teachers to be able to support students' understanding of computational concepts and their application in any subject areas. As with any skill, this form of thinking is best taught and learned in context and embedded in the disciplines being studied (Grover, 2018). Specifically, it involves methodologically equipping teachers with the necessary teaching strategies to incorporate computational thinking into their teaching and to practice it in meaningful ways that enable students to use fundamental concepts and principles to solve disciplinary, interdisciplinary and transdisciplinary problems.

Computational thinking helps teachers in all disciplines to plan, address and improve student learning outcomes. Research suggests that students exposed to computational thinking show significant improvements in problem solving, critical thinking (Calao et al., 2015), logical (logical organisation and data analysis) and systemic thinking. The role of computational thinking in education has been highlighted by many researchers (Nardelli, 2019), who have focused on understanding when and how to apply the essential skills associated with it. Consider, for example, the algorithms in computing and computational thinking that underpin everyday behaviours and tasks, from the simple execution of a simple cooking recipe to more complex instructions in other contexts. There is a common misconception that they are only used to solve mathematical problems and are not applicable to other disciplines. Introducing students to algorithms using examples from everyday life, such as the steps involved in dental hygiene or carrying out a laboratory experiment, provides the basis for understanding how to develop algorithms and subsequently implement computer programs.

Students can be introduced to the concept of computational thinking to learn complex processes of abstraction through the creation of models (such as that of the solar system) that help students to learn how to reduce complexity to bring an artefact back to its essence and to know what the artefact itself is.

The scope and nature of computational thinking involves learning strategies that model levels and guiding students to use them independently. Similarly, Barr and Stephenson (2011) argue that given that students will be entering a workforce heavily influenced by computing, it is essential that they begin to engage with the ideas of computational thinking, algorithmic problem solving practices and computational applications in the disciplines.

This will help them to integrate the of computational methods and tools into different areas of learning.

The transdisciplinary nature of computational thinking skills provides an opportunity to integrate them into all areas of teaching and learning objectives through a challenging effort that needs to be undertaken at a systemic level, as is the case with the debate on their inclusion in STEM subjects.

Hemmendinger (2010) stated that it is important to teach students to think like a "computer scientist", "economist", "physicist", "artist" and so on, in order to understand how to use computation to solve problems related to specific disciplines and to develop new questions that can be fruitfully explored. Despite the current lack of clarity about the definition of computational thinking, current studies provide a good starting point for conceptualising it.

Computational theories, information technologies and algorithms therefore play a key role in education in defining the framework of skills that students and teachers are expected to acquire, but at the heart of the relationship between concepts, practices and computational arrangements we find problems (Figure 2).

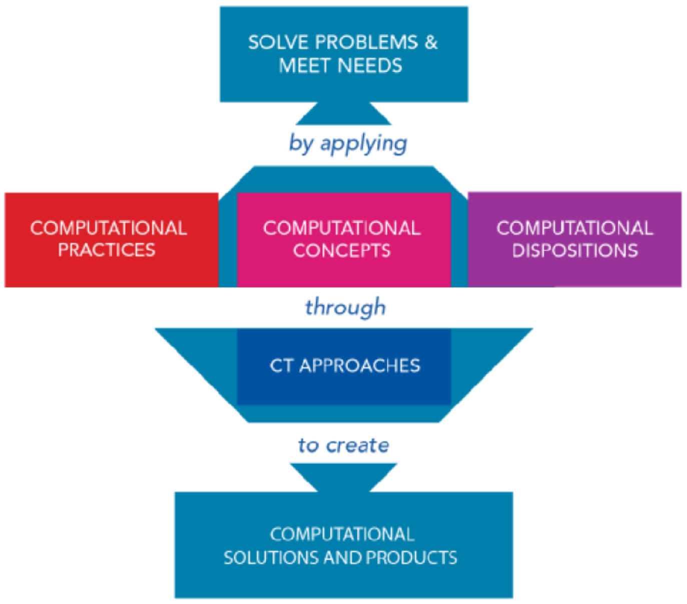
SOLVE PROBLEMS & MEET NEEDS
by applying
COMPUTATIONAL PRACTICES
COMPUTATIONAL CONCEPTS
COMPUTATIONAL DISPOSITIONS
through
CT APPROACHES
to create
COMPUTATIONAL SOLUTIONS AND PRODUCTS

Figure 1. Let's Talk Science Computational Thinking Literature and Curriculum Review (2018)

6. Computational Thinking and Teacher Training

The lack of preparation of teachers in initial and in-service training in computational thinking requires a reflection on the skills needed to enable them to help their students acquire and use this form of thinking. However, to achieve this, it is necessary to prepare teachers to incorporate computational thinking skills into their training and teaching of their subject, in order to activate teaching practices that can guide their students to use computational thinking strategies. As mentioned above, researchers in the field have argued that computational thinking must be considered on a par with basic skills, i.e. literacy and numeracy, in order for all students to understand how the world works. Education must also address the development of knowledge and skills related to computing, which is now so integrally intertwined with every profession (Grover, 2018).

However, there is still too little research on how to prepare teachers to integrate computational thinking into their training, as there is very little understanding of how to avoid overlap between the content areas of computing, computational thinking and technologies. This complication is compounded by the fact that there are few teacher training institutions that offer specific programmes and forms of certification in appropriate computational skills.

However, computational thinking requires teachers to have basic concepts and skills that enable them to break down, abstract, recognise patterns, use algorithmic and logical thinking (which involves reasoning), measure and evaluate, debug (which involves finding and correcting errors in the operation of a system or programme), collect and present data.

The first step, which we would argue is essential, is for trainers in initial teacher education programmes to address this absence by introducing specific objectives related to the processes of acquiring computational thinking. Policy makers and teacher educators, who are called upon to define the skills needed for teaching, should also consider the importance of enabling future teachers to think computationally and then be able to teach it to their students. This requires them to have a deep awareness of their own computational knowledge and how it relates to what students learn in the classroom. In this context, Darling-Hammond and Bransford (2005) have put forward a training proposal that could be adapted to prepare teachers to integrate computational thinking in the classroom. This also builds on the emerging impetus from governments around the world to expand opportunities for learning computer science in primary and secondary schools, including higher education (Czerkawski & Lyman, 2015).

Another suggestion is to introduce computational thinking training into existing educational technology courses by redesigning them within teacher training programmes, with the aim of incorporating algorithms into a writing activity (e.g. asking students to write a detailed recipe with step-by-step instructions for making a favourite dish, incorporating data analysis and pattern recognition by having students collect and analyse population statistics and use them to identify and present trends, etc.).

For teachers, embedding computational thinking would help them to meet students' learning goals and improve the quality of teaching and learning by equipping them with those data analysis tools that could enable students to present data and information through infographics and advanced tools such as Google Charts to present data dynamically through customisable graphs, and it would allow them to explore the basic ideas that are at the heart of computational thinking with the intention of making it understandable to students.

7. Computational Thinking to Learn to Teach

Compared to teachers in initial training, computational thinking in the educational context gives them the opportunity to better guarantee a set of tools for problem solving, offering a frame of reference around which the experience of primary and secondary school teachers could be modelled. The incorporation of skills related to the construction of computational thinking in teaching and learning becomes an alphabetical problem in training, offering teachers the opportunity to understand how to incorporate abstraction into specificity. Similarly, teachers could learn to incorporate computational thinking into lesson plans to enable students to collect and integrate data/information from multiple sources to visually represent common objects; to incorporate data collection, analysis and representation into any activity; to collect data and identify and represent patterns in that data; to explore how to use large data sets to identify patterns; and to discuss the implications of increasing access to large amounts of personal data. National and international training courses for existing teachers already provide opportunities to introduce teachers to the breadth of concepts and skills in computational thinking, and to engage with tools that support the development of specific skills.

However, this dimension is still a nuanced presence in the curriculum and in training policies, which would rather be fundamental to ensure the translation of what is exposed in educational contexts in order to ensure the quality of education with a view to the future.

Research

The preliminary exploratory research aimed to understand teachers' views on the inclusion of computational thinking in teacher education curricula, and to understand whether such a way of thinking could help teachers achieve various educational goals and improve the quality of teaching and learning. It was conducted with 28 primary school teachers (26 women and 2 men), aged between 28 and 48, who were interviewed using a semi-structured interview technique in September 2023.

The interviewees came from schools located in the central region of Italy and had previous experience of training in computational thinking.

There were three interview sessions for each participant, each lasting a total of 50 minutes, in order to avoid overburdening the interviewees and to allow for a sufficiently in-depth analysis of the phenomena studied.

Questions

The questions in the research plan provided an in-depth overview of teachers' perceptions, experiences and needs regarding the integration of computational thinking in school curricula, teacher training pathways and teaching-learning processes. The questions helped to gather valuable information about teachers' experiences, opinions and perceptions of students' and teachers' training in computational thinking, and provided a solid exploratory basis for initiating a possible large-scale investigation. However, in this paper, due to space limitations, the clustering of the participants' responses refers only to the questions related to the benefits of incorporating computational thinking into the school curriculum.

In order to process the responses, the NVivo software was used as the main tool for qualitative analysis. Its functions made it possible to carry out a data coding procedure, allowing the assignment of "labels" to represent the concepts identified based on their own characteristics and affinities between the data. This process resulted in categories, which are more abstract groupings of deeper and more complete concepts and results.

Questions & Raises

Dimensions	**Factors**	**Questions**	**Raises**
Socio-demographic information	**Information data**	**Age**	
		Type	
		Qualification	
		Years in the role	
		Working experience	
		Area of residence	
Training and skills	**Continuous training and professional development**	What type of refresher courses or training on computational thinking have you attended?	Can you indicate how many courses have you attended?
			Can you describe your experience?

	Technology and digital skills	How would you rate your level of digital competence?	
		What teaching technologies do you use regularly?	
	Preparation and training	How prepared do you feel to teach computational thinking?	
		What types of training and support do you think are necessary to effectively integrate computational thinking into teaching practice?	
Perceptions, experiences and benefits	**Perception of computational thinking**	What is your idea of computational thinking?	Can you give a definition of computational thinking?
		Do you think it is useful to incorporate computational thinking into teaching?	How do you think it is useful to incorporate computational thinking into teaching?
	Understanding and attitude	Do you think it is useful to train students in computational thinking?	
	Previous experiences	Have you had any past teaching experience where computational thinking was concerned?	If you have had experience, can you describe an example of an activity or project that involved computational thinking? Have you had any instructional planning experiences that involved integrating computational thinking into lesson plans?
			What were your

			impressions and/or results achieved?
			What kind of feedback have you received from students regarding activities related to computational thinking?
	Perceived benefits	What do you think are the benefits of incorporating computational thinking into school curricula?	
		What concrete benefits do you think students can derive from training in computational thinking?	How might it impact their academic success and professional future?
		In your opinion, how does computational thinking impact student motivation and interest?	
		What do you think are the main challenges that teachers encounter in integrating computational thinking into teaching?	
		What do you think could be the barriers or impediments to the development of computational thinking at school?	
Incorporation	**Incorporation into curricula**	Do you believe it is necessary to incorporate computational thinking into student training curricula?	
		What tools or resources do you think would be useful for integrating computational thinking into school curricula?	
		To what extent do you believe that computational thinking can help achieve various educational objectives?	

		What types of educational goals do you think computational thinking can support (e.g. developing problem solving skills, improving creativity, higher order skills, etc.)?	
		How do you think integrating computational thinking can help achieve educational goals such as problem solving, creativity and critical thinking?	
		How do you think computational thinking can contribute to the development of students' digital skills?	
		Do you think computational thinking can improve students' digital skills?	If yes, in what way?
	Incorporation into teacher training programs	Do you believe there is a need to incorporate computational thinking into initial teacher training courses?	How do you think computational thinking could be incorporated into initial teacher education programmes? What are the key elements that should be included in these pathways?
		What tools or resources do you think would be useful for integrating computational thinking into initial teacher education pathways?	
		Do you think teacher education programs currently adequately include computational thinking in their curricula?	Why yes? Why not?
Teaching-learning processes	**Improvement of teaching-learning processes**	Do you think that training in computational thinking can improve the quality of your teaching?	Can you elaborate on how?

	Challenges and solutions	What are the main challenges you foresee in integrating computational thinking into curricula and teaching programs?	
		How do you think these challenges can be addressed?	What additional supports do you think would be useful?
	Collaboration and professional development	If your school wanted to enhance students' training in computational thinking, how could it, in your opinion, better support the integration of computational thinking?	
		What professional development opportunities do you think would be helpful in improving your skills in computational thinking?	Do you find it useful?
Future and sustainability	**Impact on the quality of teaching and learning**	What, in your opinion, could be the impact of computational thinking on the quality of teaching and learning?	
		What observable evidence (e.g. student performance, engagement, creativity) do teachers attribute to the use of computational thinking?	
	Support and collaboration support	How could schools and educational institutions better support teachers in the transition to integrating computational thinking into school curricula?	
		What types of collaboration (with colleagues, experts, institutions) do you think could be useful to facilitate the adoption of computational thinking?	
	Sustainability and long-term support	What measures do you think are necessary to ensure the long-term sustainability of the integration of computational thinking?	
		How, in your opinion, can educational institutions support	

		teachers in integrating computational thinking at school?	
	Innovations and improvements	What can we count on to improve the teaching of computational thinking?	
		How do you see the future of teaching computational thinking in schools?	
	Quality of teaching and learning	Do you believe that the inclusion of computational thinking can improve the overall quality of teaching and learning?	Can you explain how?
Observations		Do you want to add something? Reflections, additions etc.	

Results

The benefits that teachers emphasize as significant are outlined below:

1. Development of Higher Cognitive Skills

Critical thinking and problem solving: Teachers acquire tools and methods to develop students' critical thinking, analysis and complex problem-solving skills.

Abstraction skills: develops the ability to extract basic concepts from complex problems, facilitating the understanding and management of complex information.

2. Promote interdisciplinarity

Curricular integration: Teachers learn to integrate computational thinking into different disciplines, making learning more integrated and meaningful. For example, they can use coding techniques to teach maths, science, language arts and other subjects.

Connections across disciplines: Facilitates making connections between different areas of study, helping students see the relationships between seemingly separate concepts.

3. Prepare students for the future

21st Century Skills: Equipping students with essential skills for the future, such as programming, logic, data management and digital literacy, which are increasingly in demand in the job market.

Adaptability and Innovation: Teaching students to approach new and complex problems with creativity and adaptability, preparing them to become innovators and leaders in their future fields of work.

4. Improve Teaching Effectiveness

Active methods: Encourage the use of active and participatory teaching methods, such as problem-based learning, which can increase student engagement and motivation.

Immediate feedback: The use of digital and programming tools allows you to provide immediate feedback to students, improving the learning process.

5. Facilitate inclusion and differentiation

Personalised learning: Provides tools to tailor instruction to students' needs and abilities, making learning more inclusive.

Accessibility: Digital resources and learning platforms can be adapted to support students with different abilities and learning styles.

6. Strengthen the professionalism of teachers

Professional development: Provides teachers with up-to-date and relevant skills, increasing their professionalism and preparedness.

Collaboration and networking: Encourages the creation of communities of practice and professional networks where teachers can share resources, ideas and support.

7. Supporting educational innovation

Culture of innovation: supports a culture of innovation in schools, encouraging teachers to experiment with new teaching methods and technologies.

Adapting to change: Helping schools to be more flexible and responsive to technological and social change, while keeping education relevant and modern.

Incorporating computational thinking into teacher education has a significant and positive impact on different aspects of the education system, not only helping to improve teachers' teaching skills, but also preparing students to become critical thinkers, problem solvers and innovators in the rapidly changing world of the 21st century. The need for interdisciplinary links with other disciplines (pedagogy, educational psychology, etc.) remains a central aspect. In the wake of other studies (Yadav et al., 2014; 2011; Yadav & Korb, 2012), challenges to the inclusion of computational thinking in schools emerge, including the need for more institutional support and adequate resources to successfully implement it in curricula, including by adequately preparing teachers to build such critical skills in students. The implications for curriculum design and strategies to overcome resistance to change are many. In addition, teachers could take advantage of existing networks in education and computing that would allow them to collaborate on curriculum development and research into the teaching of computational thinking. This would allow computer scientists and teacher educators to work together to develop activities for implementing computational thinking in educational contexts, and to learn how to use this knowledge to teach children to think computationally in the context of specific subjects or disciplines, helping them to connect them to their everyday lives and contexts through the use of computational thinking in the classroom. Recognising the importance for teachers to acquire computational thinking and therefore to be able to count on adequate training in this sense in their curricula. Several national and international organisations, such as the International Society for Technology in Education (ISTE), the Italian Association for Information and Automatic Calculation (AICA), etc., are working to develop and share tools and resources for teachers, such as Google's Exploring Computational Thinking website, which offers examples of lesson plans and curricula, and a collection of videos that show how it is possible to use concepts related to computational thinking, providing courses for teachers at all levels.

These resources provide a valuable starting point for supporting teachers in their training and with what they need.

8. Conclusions

Although computational thinking has been consistently identified as a critical skill needed by students and teachers in the 21st century, and the educational world has shown a growing interest in promoting its development in recent years, it is seen as a way of thinking whose skills, dispositions and attitudes should become an integral part of school curricula, as it is considered one of the primary skills, along with reading, writing and arithmetic. In practice, this can be difficult to achieve, in part because of an incomplete understanding of what it is and its potential relationship to other skills, competencies and cognitive characteristics of individuals, as well as its ability to be improved through education using precise teaching strategies. The 21st century is heavily influenced by computing, making it imperative that teachers integrate computational thinking into primary and secondary education and prepare themselves to teach these higher-order skills to students. Training programmes are the best opportunity to engage teachers early in their preparation to formulate ways to integrate computational thinking into their practice. However, this effort should involve collaboration between curricular, IT, and support teachers, as each of them brings complementary skills to the development of teacher education.

Today, computational thinking seems to be a useful skill for everyone because, as Wing argued in 2006, "ubiquitous computing was yesterday's dream that is becoming today's reality; computational thinking is tomorrow's reality" (Wing, 2006, p. 34) and it is not a way to make people more like computers, but rather to enable them to use them effectively to solve the problems of the computer age as it changes the way we think and has become an integral part of the functions of everyday life.

Acknowledgments

We would like to thank all the teachers who took the time to participate in this study.

Authors contributions

Prof. Antonella Nuzzaci was responsible for study design and revising responsible for data collection.

Funding

Not applicable.

Competing interests

Not applicable.

Informed consent

Obtained.

Ethics approval

The Publication Ethics Committee of the Macrothink Institute.

The journal's policies adhere to the Core Practices established by the Committee on Publication Ethics (COPE).

Provenance and peer review

Not commissioned; externally double-blind peer reviewed.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

Data sharing statement

No additional data are available.

Open access

Copyrights

References

Armoni, M. (2016). Computing in schools: Computer science, computational thinking, programming, coding: The anomalies of transitivity in K-12 computer science education. *ACM Inroads*, *7*(4), 24-27. https://doi.org/10.1145/3011071

Baird, W. E., & Borich, G. D. (1987). Validity considerations for research on integrated science process skills and formal reasoning ability. *Science Education*, *71*(2), 259-269. https://doi.org/10.1002/sce.3730710212

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: a digital age. *Learning & Leading with Technology*, *38*(Mar./Apr.), 20-23.
https://api.semanticscholar.org/CorpusID:53794701

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

Bitner, B. L. (1991). Formal operational reasoning modes : Predictors of critical thinking abilities and grades assigned by teachers in science and mathematics for students in grades nine through twelve. *Journal of Research in Science Teaching, 28*(3), 265-274. https://doi.org/10.1002/tea.3660280307

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing, 1*(2), 67-69. https://bit.ly/3sPUHaz

Calao, L. A., Jesús Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with Scratch: An experiment with 6th grade students. In *Proceedings of the Design for Teaching and Learning in a Networked World 10th European Conference on Technology Enhanced Learning* (Toledo, Spain, Sept. 15–18) (pp. 17-27). Springer International Publishing. https://doi.org/10.1007/978-3-319-24258-3_2

Chambers, J. (1988). Teaching thinking throughout the curriculum Where else? *Educational Leaclership, 45*(7), 4-6.

Chandron, S., Treagust, D. S., & Tobin, K. (1987). The role of cognitive factors in chemistry achievement. *Journal of Research in Science Teaching, 24*(2), 145-160. https://doi.org/10.1002/tea.3660240207

Cooper, S., Forbes, J., Fox, A., Hambrusch, S., Ko, A., & Simon, B. (2016). The Importance of Computing Education Research. White paper, *Computing Community Consortium, Jan. 14,* 1-12.
http://cra.org/ccc/wp-content/uploads/sites/2/2015/01/CSEdResearchWhitePaper2016.pdf

Corradini, I., Lodi, M., & Nardelli, E. (2018). An investigation of Italian primary school teachers' view on coding and programming. In Proceedings of the Informatics in Schools: Fundamentals of Computer Science and Software Engineering (ISSEP). S. N. Pozdniakov & V. Dagiene (Eds.), *Lecture Notes in Computer Science* (Vol. 11169, pp. 228-243). Cham: Springer. https://doi.org/10.1007/978-3-030-02750-6_18

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. *Computing at School Community*, 1-18. https://community.computingatschool.org.uk/resources/2324.

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends, 59*(2), 57-65. https://doi.org/10.1007/s11528-015-0840-3

Dalziel, J. (2010). *Practical e teaching strategies for predict–observe–explain, problem based learning and role plays*, Sydney: LAMS International.

Dalziel, J., & Dalziel, B. (2011). Adoption of learning designs in teacher training and medical education: Templates versus embedded content. In L. Cameron & J. Dalziel (Eds.), *Proceedings of the 6th International LAMS & Learning Design Conference 2011: Learning design for a changing world* (pp. 81-88). Sydney: LAMS Foundation.

d'Angelo, E. (1971). *The teaching qf critical drinking*. Amsterdam B.R. Gruner N.V.

Darling-Hammond, L., & Bransford, J. (Eds.) (2005). *Preparing Teachers for a Changing World: What Teachers Should Learn and Be Able to Do.* San Francisco: Jossey-Bass.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications of the ACM, 52*(6), 28-30. https://doi.org/10.1145/1516046.1516054

Denning, P. J. (2013). The science in computer science. *Communications of the ACM, 56*(5), 35-38. https://doi.org/10.1145/2447976.2447988

Denning, P. J., Tedre, M. (2021). Computational Thinking: A Disciplinary Perspective. *Informatics in Education, 20*(3). https://doi.org/10.15388/infedu.2021.21

Denning, P. J., & Tedre, M. (2019). *Computational Thinking.* Cambridge, MA: The MIT Press, Cambridge. https://doi.org/10.7551/mitpress/11740.001.0001

Espinal, A., Vieira, C., & Magana, A. J. (2024). Professional Development in Computational Thinking: A Systematic Literature Review. *ACM Transactions on Computing Education, 24*(2), 1-24. Article 27. https://doi.org/10.1145/3648477

Facione, P. A. (2007). *Thinking and Reasoning in Human Decision Making.* Millbrae, CA: The California Academic Press.

Facione, P. A. (2011). *Think Critically.* Englewood Cliffs, NJ: Pearson Education.

Franklin, D. (2015). Putting the computer science in computing education research. *Communications of the ACM, 58*(2), 34-36. https://doi.org/10.1145/2700376

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK schools*. London, UK: The Royal Society.
http://royalsociety.org/education/policy/computing-in-schools/report/.

Grover, S. (2018). *The 5th 'C' of 21st century skills? Try computational thinking (not The K-12 Educational Technology Handbook 15 coding*. EdSurge News: https://edtechbooks.org/-Pz.

Grover, S., & Pea, R. D. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42(*1), 38-43. https://doi.org/10.3102/0013189X12463051

Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads, 1*(2), 4-7. https://doi.org/10.1145/1805724.1805725

Inhelder, & Piaget, P. (1955). *De la logique de l'enfant à la logique de l'adolescent: essai sur la construction des structures opératoires formelles*. Paris: Presses Universitaires de France.

Jenkins, H., Purushotma, R., Weigel, M., Clinton, K., & Robison, A. J. (2010). *Culture partecipative e competenze digitali. Media education per il XXI secolo.* Milano: Guerini.

Jonassen, D. H. (2008). Instructional design as design problem solving: an iterative process. *Educational Technology, 48*(3), 21-26. http://www.jstor.org/stable/44429574

Kahney, H. (1993). *Problem Solving: Current Issues*. Buckingham, U.K.: Open University Press.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Let's Talk Science (2018). *Let's Talk Science Computational Thinking Literature and Curriculum Review*. London, CA-ON: Let's Talk Science.

Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about Computational Thinking. *ACM SIGCSE Bulletin, 41*(1), 260-264. https://doi.org/10.1145/1539024.1508959

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, 41*(4), 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Nardelli, E. (2019). Do We Really Need Computational Thinking? *Communications of the ACM,* 62(2), 32-35. https://api.semanticscholar.org/CorpusID:146145154. https://doi.org/10.1145/3231587

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

Piaget, J. (1936). *La naissance de l'intelligence chez l'enfant*. Neuchatel: Delachaux et Niestlé.

Piaget, J. (1947). *La psychologie de l'intelligence*. Paris: Colin.
https://doi.org/10.4324/9780203278895

Repenning, A., Basawapatna, A., Escherle, N. (2017). Principles of Computational Thinking Tools. In P. J. Rich & C. B. Hodges, *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 291-305). London: Springer. https://doi.org/10.1007/978-3-319-52691-1_18

Resnick, M., & Rusk, N. (2020). Coding at a crossroads. *Communications of the ACM, 63*(11),120-127. https://doi.org/10.1145/3375546

Riley, D., & Hunt, K. A. (2014). *Computational thinking for the modern problem Solver*. New York: Chapman and Hall/CRC. https://doi.org/10.1201/b16688

Roberts, R., & Gott, R. (1999). Procedural understanding: its place in the biology curriculum. *School Science Review, 81*(294), 19-25.

Roberts, R., & Gott, R. (2000). Procedural understanding in biology: how is it characterised in texts? *School Science Review, 82*(298), 83-91. https://doi.org/10.1126/science.298.5591.82

Selby, C., & Woollard, J. (2013). *Computational thinking: the developing definition.* University of Southampton. http://eprints.soton.ac.uk/356481.

Swartz, R. J., & Perkins, D. N. (1990). *Teaching thinking: Issues and approaches*. Pacific Grove, CA: Midwest Publications.

Tamir, P., Stavy, R., & Ratner, N. (1998). Teaching science by inquiry: assessment and learning. *Journal of Biological Education, 33*(1), 27-32. https://doi.org/10.1080/00219266.1998.9655633

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715-728. https://doi.org/10.1007/s10639-015-9412-6

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions A, 366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Yadav, A., & Korb, J. T. (2012). Learning to teach computer science. *Communications of the ACM, 55*(11), 31. https://doi.org/10.1145/2366316.2366327

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 14*(1), 1-16. https://doi.org/10.1145/2576872

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, Mar. 9-12) (pp. 465-470). New York, NY: ACM Press. https://doi.org/10.1145/1953163.1953297