Towards a Consensus about Computational Thinking Skills: Identifying Agreed Relevant Dimensions

Bostjan Bubnic
Faculty of Electrical Engineering and Computer Science
University of Maribor
b.bubnic@gmail.com

Tomaz Kosar
Faculty of Electrical Engineering and Computer Science
University of Maribor
tomaz.kosar@um.si

Abstract
Research on Computational thinking (CT) has already entered its second decade, but still lacks a clear definition that researchers would agree upon. There are even suggestions that the definition of CT is not indispensable and that researchers should focus on other aspects, such as how to include CT in courses, curriculums and how to observe the acquisition of CT. However, it is generally agreed that CT is an important skill within the computer science, while it also extends beyond computing as being a fundamental skill for problem solving in all scientific and engineering disciplines. Moreover, there is a great interest of researchers and educators to explore how to include CT in K-12 context. Our study builds upon the consensus, that multiple skills are involved in CT. Based on the literature review, this study tries to identify a basic, domain independent dimensions of CT that researchers agree upon. The results of this study identify abstraction and algorithms as relevant, domain independent dimensions to build upon consensus. We hope that the study results will encourage further research towards consensus about general, domain independent set of skills that forms CT. This would be particularly beneficial in assessing and teaching CT.

1. Introduction
The phrase computational thinking (CT) was introduced by Seymour Papert in his seminal book Mindstorms: Children, Computers, and Powerful Ideas (Papert, 1980, p.182). His main objective for teaching Logo, an educational programming language, was to improve the student's ability to think procedurally. Lately, CT has seen enormous growth in popularity, after it was reintroduced and popularized by Wing in 2006. Until recently, computing has been considered as a domain specific set of problem solving skills possessed by computer scientists, mathematicians and engineers. However, in a seminal article published in 2006, Wing described CT as a way of "solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science" (Wing, 2006, p.33). She exposed the ability to think recursively, the use of abstraction and decomposition as important skills when dealing with complex tasks. She also argued that "computational thinking is a fundamental skill for everyone, not just for computer scientists" (Wing, 2006, p.33). Since then, CT has received increasing attention among researchers and practitioners within the field of education. A development of general-purpose problem solving skills across the curriculum is the primary aim of integrating CT into various national curriculums.

As CT originates from computer science and programming, the common approach to learn and assess CT today is by educational programming languages like Scratch, Alice or Snap! It is widely agreed that programming is a useful, but domain specific skill, while CT is applicable to wider educational spectrum and in everyday life. The shift from computing to the general problem solving domain has stimulated researchers to investigate a broader aspects of CT. Recently, CT research in science, technology, engineering and mathematics (STEM) disciplines emerge as prevalent (Khine, 2018). Furthermore, there are several researches exploring CT within non-STEM disciplines, such as language lessons (Sabitzer et al., 2018) and music notation (Barate et al., 2017). Broader CT spectrum presents great opportunities but also challenges for educational researchers, primary and secondary level educators as well as computer science educators. To determine effective methods for teaching, learning and assessing CT, a definition of CT and its scope is needed. Moreover, the main components of CT need to be identified.

To date, there is no consensus concerning the definition of computational thinking. In addition to different perspectives about the definition, there is even a disagreement about the importance of achieving consensus. On the one hand, researchers are advocating the necessity of CT definition, like Selby and Woollard (2014), Werner et al. (2012) or Dong et al. (2019) who argue that "there is a need for a CT definition that teachers can effectively use to communicate CT to students in core classroom settings" (Dong et al., 2019, p.906). On the other hand, researchers argue that the focus should be on other aspects, such as how to include CT in classes, curriculums and how to observe the acquisition of CT. In this context, Aho argues that "any static definition of computational thinking likely would be obsolete 10 or 20 years from now" (National Research Council 2011, p.37). CT definitions are out of scope of this study.

Apart from CT definition, researchers have begun to characterize computational thinking by means of CT taxonomies and frameworks, where particular concepts and practices were observed. Moreover, it is generally agreed that multiple skills are involved in CT. The main contribution of this paper is the identification of general, domain independent components of CT that will encourage further research towards consensus about general, domain independent components that forms CT. An additional contribution is related to unveiling the characteristics of relevant CT components with an emphases on computational, non-computational, cognitive and epistemological aspects of particular component.

2. Related Work

It is generally agreed that multiple skills are involved in CT. Wing (2008) addressed the significance of CT concepts as part of "fundamental questions: What are the elemental concepts of computational thinking?" (Wing, 2008, p.3270).

However, there is still no consensus regarding the definitive or necessary components of CT. When computational thinking was reintroduced by Wing (2006), abstraction and decomposition were the fundamental components, while heuristic reasoning, problem reformulation, recursion and systematic testing were referenced as important cognitive processes for solving problems efficiently. The initial set of components were later refined with automation (Wing, 2008). After initial component conceptualization, researchers have begun to characterize computational thinking by means of CT taxonomies and frameworks, where particular concepts and practices were observed. Brennan and Resnick (2012) proposed a programming specific CT framework, where components were associated with programming specific concepts and artefacts. Weintrop et al. (2016) constructed a computational thinking taxonomy for mathematics and science. They "narrow the scope of computational thinking away from generalities, providing a sharper definition that is distinct from computer science". (Weintrop et al., 2016, p.128). Based on literature review, Shute et al. (2017) presented CT components that are "common among researchers".

By examining the literature we found only two papers researching on achieving the consensus of the CT components. Barr et al. (2011) report on consensus that emerged regarding the essential elements of CT during the meeting of a diverse group of educators with an interest in CT from higher education, K-12 and industry. Rich and Langton (2016) used Delphi process to formalize CT definition, where CT components were also part of consensus goals.

2.1. Terminology

When addressing the multiple dimensionality of CT, researchers, educators and practitioners use the terminology interchangeably, not being consistent. Brennan and Resnick (2012) propose a CT where components are classified into computational concepts, computational practices and computational perspectives. However, the proposed computational concepts are domain specific, because they address the programming constructs only, such as sequences, loops, conditionals. The computational practices category includes problem solving practices that occur in the process of programming, such as experimenting, iterating, abstracting and modularization. On the contrary, Barr and Stephenson (2011) propose the category of computational thinking concepts that includes abstraction, decomposition and automation, being general and domain independent dimensions. While reviewing literature, we found that authors use different terms when they address CT dimensions, such as skills, practices, components, concepts or abilities and that they are not consistent in their usage. We will be using the term "CT concept" when referring to theoretical, abstract, domain independent dimensions of CT, while the term "CT practice" will be used for domain specific, practical, tangible dimensions.

Weintrop et al. (2016) report they moved from term "skill" to a broader and more actionable term "practice based" on suggestions from many teachers from STEM disciplines.

3. Literature review

Although the purpose of this work was not a systematic literature review, we tried to follow the protocol defined by Kitchenham and Charters (2007) as much as possible.

3.1. Scoping

RQ1: What are the relevant theoretical, domain independent dimensions of CT? RQ2: What is the proportion of intendent usage of relevant CT dimensions in the literature?

Answering RQ1 would enable us the examination of the widest possible scope of domain independent CT components. Additionally, the components with the highest number of occurrences could be selected as prime prospects for achieving the agreement about the definitive and necessary components of CT. Answering RQ2 would show the proportion of particular areas of interest and the scope of identified CT components within literature.

3.2. Planning and identification

To identify the broadest scope of CT aspects, we included the following terms as keywords for document screening and snowballing: CT components, CT skills, CT practices, CT dimensions, CT aspects and CT abilities.

The first step in the creation of our initial corpus of relevant papers was the review of the existing CT literature with the aim to identify the broadest scope of CT aspects. Our investigation began by investigating four literature review papers. Inductive qualitative content analysis was conducted by Kalelioglu et al. (2016). They reviewed 125 papers researching on CT to provide a framework about the notion, scope and elements of CT. Hsu et al. (2018) conducted a meta-review of CT studies published in academic journals from 2006 to 2017. Based on the analyses of various educational CT aspects, mostly focusing on K-12 domain, they generated a list of 19 CT components while providing relevant references for each component. Systematic literature review was also conducted by Lockwood and Mooney (2018) to get an overview of the work that has been carried on in the secondary education. Moreover, their objective was to identify potential gaps and opportunities that might still exist in the scope of CT education. Their literature review has identified 58 papers referring to CT definitions, course design, teaching methodology or assessing CT. Sondakh (2017) published a paper that provided the insight of CT in higher education, particularly focusing on assessing CT skills. She identified 22 relevant CT skills within 16 papers. It should be noted that the papers were selected with the intent to include the widest possible educational spectrum. Based on the detailed review of the four aforementioned literature reviews, our initial corpus of 135 relevant papers was ready for screening.

3.3. Inclusion criteria and paper screening

The inclusion criteria had to support the main purpose of this study that is to identify basic, domain independent dimensions of CT. After screening summary, introduction and discussion sections of each paper from the corpus of the initial 135 papers, the paper was included for CT concept analysis only if it was a CT definition proposal, CT taxonomy proposal or CT assessment. Moreover, the papers that applied the existing conceptual taxonomy to a specific domain or the papers that proposed a new conceptual taxonomy based on existing CT definition or existing CT taxonomy, were also included. In the second phase, CT concept analysis was performed to build a final paper corpus. Only papers that advocated conceptual dimensions were selected into final corpus. During the process, we also identified papers that built a conceptual taxonomy first that was later implemented within programming domain, where conceptual dimensions were translated to specific programming constructs. They were also included in the final corpus. A large number of papers were building on CT taxonomies specific to programming domain, but were not included in the final corpus. Among which, Brennan and Resnick (2012) was the most referenced paper. At the end of this process twenty-eight papers have been included in the final corpus.

4. Results

This section presents the results of the analysis of 28 papers that were included in the final corpus.

RQ1: What are the relevant theoretical, domain independent dimensions of CT?

The aim was to discover the widest possible range of general, domain independent CT concepts. Thirty-six distinct CT concepts were identified, with frequencies ranging from 23 to 1. CT concepts with the highest frequencies are presented in Table 1. Relevant CT dimensions are presented in CT Concept column, number of occurrences for each dimension is presented in Frequency column. The Reference column presents the list of papers where the particular dimension has been identified. It should be noted that different reference format is used to list the papers in the Reference column for greater legibility. The most common concepts are underlined: abstraction and algorithm.

As presented in Table 1, the abstraction concept and the algorithm concept have 23 occurrences within the literature. However, as can be observed from the reference column in Table 1, they were not identified within the same papers. The frequencies of other identified concepts are much lower than abstraction and algorithm.

CT Concept	Frequency	Reference
<u>Abstraction</u>	23	[3] [7] [9] [10] [15] [18] [22] [23] [27] [28] [33] [34] [38] [63] [64] [66] [70] [73] [74] [82] [83] [84] [90]
<u>Algorithm</u>	23	[3] [7] [9] [10] [13] [15] [18] [22] [23] [27] [28] [34] [38] [45] [49] [63] [64] [66] [70] [73] [74] [82] [90]
Decomposition	14	[3] [7] [18] [22] [23] [28] [38] [64] [66] [70] [73] [74] [82] [90]
Data representation	11	[7] [10] [13] [15] [27] [28] [66] [69] [73] [84] [90]
Modeling	9	[13] [15] [27] [28] [33] [34] [51] [69] [84]
Evaluation	8	[3] [13] [18] [27] [64] [69] [74] [82]
Generalization	8	[3] [18] [27] [28] [34] [64] [74] [90]

Table 1 – Identified CT concepts

For a better visualization of the frequency of CT concepts, Figure 1 shows a word cloud of all 36 distinct CT concepts.

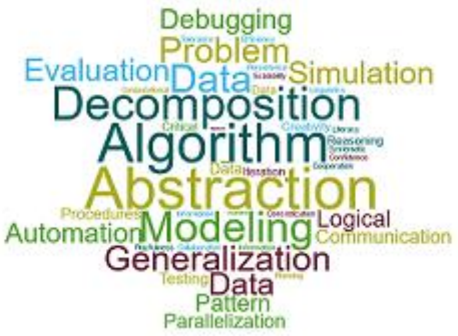

Figure 1 – Word cloud of identified CT concepts

As can be observed from Figure 1, the algorithm concept and the abstraction concept are of the same, maximum size, while other identified concepts are smaller, according to their frequency. Due to space limit, concepts with only one occurrence, such as Systematic analysis, Cooperation, Concretization, Collaboration, Information processing, Playfulness, Literacy, Computational Vocabulary, Linguistics, Scalability, Planning and Efficiency have reduced visibility.

RQ2: What is the proportion of intendent usage of relevant CT dimensions in the literature?

Table 2 presents the frequency of intendent usage of relevant CT dimensions identified in the literature. The most common identified purpose was CT definition, followed by CT assessment and CT taxonomy.

Intendent usage	Frequency	Reference
Definition	13	[7] [9] [10] [15] [18] [28] [33] [34] [49] [51] [70] [74] [82]
Assessment	10	[3] [13] [22] [27] [45] [63] [66] [69] [83] [90]
Taxonomy	5	[23] [38] [64] [73] [84]

Table 2 – Intendent usage of CT dimensions

5. Relevant computational thinking concepts

In this section we present the characteristics of relevant computational thinking concepts that were identified within our study. The concepts and their frequencies were presented in Table 1. Frequencies and paper references of intendent usage are presented at the end of description of each CT concept. However, this is by no means an exhaustive literature review of particular concepts, but rather an overview of their characteristics, scope and usage.

5.1. Abstraction

Abstraction can generally be characterized as the conceptual process of eliminating specificity by ignoring certain features. From a human ability perspective, abstraction is comprised of two complementary concepts: removing details to build simplification and deriving generalizations to illuminate essentials. Abstraction has been the focus of many studies within various disciplines, such as philosophy, psychology, cognitive science, mathematics and computer science. Whistler (2016) cites prominent philosophers who were debating how a reason evolves as an abstraction of thought. Abstract thinking was defined as the ability "to realistically imagine a problem and a solution" by developmental psychologist Piaget (1950). Piaget argued that children develop the ability to think abstractly, systematically and hypothetically as part of their last stage of cognitive development. Kramer (2007) asserted that abstraction is "fundamental to mathematics and engineering in general, playing a critical part in the production of models for analysis and in the production of sound engineering solutions" (Kramer, 2007, p.40). Frorer et al. (1997) addressed the complexity and various faces of abstraction in mathematics. Moreover, diSessa (2018) interprets mathematics as a "queen of abstract sciences". diSessa clarifies distinction between the mathematical (inferential) abstraction, abstraction in physic (empirical abstraction) and abstraction in computer science (practical abstraction). diSessa advocates that the skill of "peeling away" irrelevant particulars is not needed within the mathematical abstraction. This claim is further supported by Colburn and Shute (2007) who advocate that abstraction in "computer science is to be sharply distinguished with abstraction within mathematics". Abstraction is closely related to the modeling concept and the concept of generalization. The model is abstraction of a real or a conceptual complex system. Empirical sciences are normally concerned with two kinds of models: concrete models in the form of experimental apparatus and abstract model in the form of mathematics. The model is designed to expose significant features and characteristics of the system that is intended for the study, for the prediction or for the modification. Thus, a particular model represents only some of the many aspects of the complex system that could potentially be modeled. Namely, the aspects and the complexity of the particular model depend on the relevant level of abstraction that was employed by the model developer. Prusinkiewicz (1998) discusses the relevancy of abstraction levels in natural sciences. Moreover, Nielsen (2018) addresses the right degree of abstraction in the context of scientific knowledge and scientific abstraction. Nielsen argues that abstraction is equally as important as the scientific method, although it is relatively neglected in the science itself and in contemporary analytical philosophy of science. Computer science relates to various entities characterized as abstract or various activities characterized as abstraction, such as abstract data types, data abstraction, procedural and control abstraction. Lowe and Brophy (2017) claim that abstraction in computer science has taken on two different usages: the first use of abstraction relates to inheritance and uses the fewer details as an extension point for future functionality, while the other common use of abstraction is to hide away the details which are not required to understand a greater concept. The latter one is often conceptualized as encapsulation. This is further supported by Colburn and Shute

(2007) who advocate that information hiding is the primary objective of abstraction in computer science practices, such as programming languages, operating systems, network architecture, and design patterns. Within software engineering, abstraction involves the extraction of properties of an object according to some focus: only those properties are selected which are relevant with respect to the focus (Czarnecki, 1988). Michaelson (2017) presents several examples of programming practices concerning abstraction, such as the procedural abstraction, the functional abstraction and abstractions within arrays and recursion. Moreover, Michaelson links these programming practices with CT concepts. According to Wing (2011), abstraction process is "the most important and high-level thought process in computational thinking". Additionally, abstraction "is used to let one object stand for many. It is used to capture essential properties common to a set of objects while hiding irrelevant distinctions among them" (Wing, 2011). In the context of CT, layers of abstraction allows a problem solver to focus on one layer at the time, to define each layer and to illuminate relationships between layers (Wing, 2008). Furthermore, Csizmadia et al. (2015) define abstraction in the context of CT as the process of making an artefact more understandable through reducing the unnecessary detail. The core practice in CT abstraction is in choosing the right detail to hide that the problem becomes easier, without losing anything that is important. A key part is in choosing a good representation of the system. Different representations make different things easy to do (Csizmadia et al., 2015). Selby and Woollard (2014) cite various authors that observe abstraction in the context of developing the CT definition. Based on the literature review, they advocate that abstraction is the core concept that should be included in CT. Results of our analysis show 11 occurrences in CT definition category ([7] [9] [10] [15] [18] [28] [33] [34] [70] [74] [82]), 5 occurrences in CT taxonomy category ([23] [38] [64] [73] [84]) and 7 occurrences in CT assessment category ([9] [22] [27] [63] [66] [83] [90]).

5.2. Algorithms

We define algorithms as procedural building blocks of a computer programming, of a human thought and of a general problem solving. The purpose of this generalized definition is to manifest the multi-dimensionality of an algorithm concept. Within programming, algorithms are predefined procedures that are encoded with programming artefacts to incorporate sequencing, branching and looping. In the theoretical computer science, an algorithm is defined as a procedure that satisfies the criteria of finiteness, input, output, effectiveness, and definiteness (Knuth, 1997). In this context, algorithms are effectuated with formally defined (programming) languages, where statements are rigorously unambiguous. The aim is that such an algorithm becomes a machine executable artefact, characterized by Denning (2017) as "not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgment" (Denning, 2017, p.33). Finally, Aho (2011) provides a definition of an algorithm in the context of Turing computability theory, stating that "The Turing machine provides a precise definition for the term algorithm: an algorithm for a function f is just a Turing machine that computes f." (Aho, 2011, p.5). Mathematical algorithm is a computational procedure, a set of steps that takes an input, perform a finite number of allowed operations and produces an output. Among all the definitions of algorithms within mathematics, we emphasize the definition introduced by Fetzer (2001) that cites Kleene (1967), who defines algorithms as "effective procedures, which are rules or methods whose application to a problem within an appropriate domain leads invariably to a definite solution within a finite number of steps". Generally, the algorithmic nature of mathematics can be observed through proving many theorems that provide a finite procedure to answer a question or to calculate something (Basu et al., 2006). Algorithms are of great interest to many philosophers that advocate the relation of algorithms to cognition (Hill, 2015). In the philosophy of mind, the mental algorithm is a fragment in the computational theory of mind that treats human mind as information processing system. Fetzer (2001) cites prominent cognitive scientists and philosophers while he observes the concept of mental algorithm within computational and non-computational conception. Moreover, computational theory of mind is the foundation for cognitive theory proposed by Marr (2010), where cognitive processes consist of three layers: computational (semantic) level, algorithmic (syntax) level and implementational (physical) level. The algorithm design skills and effective usage of algorithms are fundamental practices in algorithmic problem solving. In educational context, these practices are advocated as algorithmic thinking. Knuth (1985) observed the concept of algorithmic thinking in mathematics, while Fustschek (2006) associates algorithmic thinking as one of the most important competencies that can be acquired by learning informatics. Fustschek further defines algorithmic thinking as a pool of abilities that are concerned with the ability to analyze the given problems, the ability to specify a problem precisely, the ability to construct the correct algorithm, the ability to think about all possible circumstances to a given problem and the ability to improve the efficiency of an algorithm. Moreover, Fustschek advocates a strong creative aspect of the algorithmic thinking associated with the construction of new algorithms that solve new problems. In the context of CT, algorithmic thinking is concerned with getting to a solution through a clear definition of steps. It is the ability to think in terms of sequences and rules as a way of problem solving or understanding situations (Csizmadia et al., 2015). CT is associated with algorithmic thinking, pattern recognition and generalization in solving problems when repeatable patterns can be observed. According to our analysis, algorithms have 11 occurrences in CT definition category ([7] [9] [10] [15] [18] [28] [34] [49] [63] [70] [74]), 4 occurrences in CT taxonomy category ([23] [38] [64] [73]) and 8 occurrences in CT assessment category ([3] [13] [22] [27] [45] [63] [66] [90]).

5.3. Decomposition

Decomposition deals with breaking down a problem into smaller, more manageable components where each component can be managed independently. Problem decomposition practice exists in general scientific domain. Decomposition was advocated by Polya (1957) within mathematics as a part of his problem solving techniques called heuristics. Examples are to be found in the physics, where objects are decomposed into entities of hierarchy, like molecules, atoms and subatomic particles. In discrete mathematics, modular decomposition is a technique in several domains of combinatorics, such as the graphs and hypergraphs (Habib and Paul, 2010). In computer science, distinct variants of decomposition can be observed. Parnas (1972) investigated hierarchical decomposition in the context of modularity in order to decompose complex information system into a number of smaller, manageable modules. Moreover, decomposition is the crucial part of structured, object-oriented and functional programming paradigms. In terms of CT, Wing (2008) linked abstraction layers with hierarchical decomposition. Barr and Stephhenson (2011) argue that decomposition is about breaking problems down into smaller parts that may be more easily solved. Decomposition is a way of thinking about problems, algorithms, artefacts, processes and systems in terms of their parts. These parts can then be understood, solved, developed and evaluated separately. This makes complex problems easier to solve and large systems easier to design (Curzon et al., 2014). Results of our analysis show there are 6 occurrences in CT definition category ([7] [18] [28] [70] [74] [82]), 4 occurrences in CT taxonomy category ([23] [38] [64] [73]) and 4 occurrences in CT assessment category ([3] [22] [66] [90]).

5.4. Data management

Data collection, data analysis and data representation are important aspects of the scientific research. However, in the context of CT, data is supporting computational thinking process by providing important information in the means of finding a data source, analyzing the data or using the data structures to represent data, such as graphs, charts, words or images (Conery et al., 2011). In our analysis, data representation concept has 4 occurrences in CT definition category ([7] [10] [15] [28]), 2 occurrences in CT taxonomy category ([73] [84]) and 5 occurrences in CT assessment category ([13] [27] [66] [69] [90]).

5.5 Modeling

Modeling is a core practice in science and a central part of scientific literacy. Schwarz et al. (2009) define scientific model as a representation that abstracts and simplifies a system by focusing on key features to explain and predict scientific phenomena. Working with scientific models involves constructing and using models, as well as evaluating and revising them. The development of sufficiently useful models typically requires a series of iterative "modeling cycles" where results of constructions and explanations activities are tested and revised repeatedly (Lesh and Harel, 2003). Modeling is particularly important in mathematics and engineering. Mathematical models are distinct from other categories of models mainly because they focus on structural characteristics of systems they describe, while models in physics, biology, or arts do not build upon the structural characteristics (Lesh and Harel, 2003). Kramer (2007) argues that modeling is the most important engineering technique, as models assist engineers to understand and analyze large and complex systems. Within computing, modeling is an important and common form of representing different areas and levels of informatics and software engineering. In the theoretical computer science Aho (2011) defines a model of computation as "mathematical abstraction of a computing system" (Aho, 2011, p.4). Furthermore, he emphasizes Turing machine as the most important model of sequential computation. However, modeling within computing might not be confused with computational modeling, which is the use of computers to simulate and study the behavior of complex systems using mathematics and computational science. Computational modeling can be applied to any scientific domain, particularly in empirical sciences, such as physics, neurobiology, psychology and cognitive science (Stacewicz and Wlodarczyk, 2010). Sabitzer and Pasterk (2015) observed modeling in the context of education, where competencies needed for the modeling process like abstraction, reduction, simplification, classification and generalization should be part of general education covering children of all ages. STEM educational researchers at K-12 level (Weintrop et al., 2016) have proposed CT taxonomy for mathematics and science, where, among other CT concepts, modeling and simulation concept is defined as designing, constructing, using and assessing computational models. According to our analysis, modeling concept has 5 occurrences in CT definition category ([15] [28] [33] [34] [51]), 1 occurrence in CT taxonomy category ([84]) and 3 occurrences in CT assessment category ([13] [27] [69]).

5.6. Evaluation

Evaluation generally refers to the process of determining the worth, merit or value of the subject that is exposed to evaluation – evaluand. The evaluation process normally involves some identification of relevant standards of worth, merit or value. Moreover, the process is also concerned with some investigation of the performance of the evaluands on these standards (Shaw et al., 2013). Distinction should be concerned between the term "assessment" and the term "evaluation" within educational domain. Sultana (2016) defines evaluation as the process for determining the success level of an individual based on data, while assessment is defined as the procedures or techniques used to collect the aforementioned data. In the context of CT, Curzon et al. (2014) propose evaluation as the process of ensuring an algorithmic solution is a good one: that it fits for purpose. Various algorithmic properties need to be evaluated including correctness, speed, whether they are economic in the use of resources and whether they are easy for people to use and to promote. There is a specific and often extreme focus on attention to detail in computational thinking based evaluation (Curzon et al., 2014). In our analysis, evaluation concept has 3 occurrences in CT definition category ([18] [74] [82]), 1 occurrence in CT taxonomy category ([64]) and 4 occurrences in CT assessment category ([3] [13] [27] [69]).

5.7. Generalization

Generalization can be observed from different aspects. In psychology, generalization is described as "the occurrence of relevant behavior under different, non-training conditions" (Stokes and Baer, 1977, p.350). From educational point of view, generalization is one of the fundamental activities in teaching and learning of mathematics (Hashemi et al., 2013). In software engineering, generalization is a crucial practice in object oriented programming, where generalization is the act of capturing similarities between classes and defining the similarities in a new, generalized class, called superclass. The ability to generalize and to abstract is an important concept in problem solving. Kamarudin et al. (2016) argue that generalization and abstraction provide more freedom in the construction of an idea than using a specific problem solving approach. Both concepts enable designers to initiate early moves in generating various ideas for problem solving. Within CT, generalization is associated with identifying patterns, similarities and connections. It is a way to quickly solve new problems based on previous solutions to problems, and a way of building on prior experience (Csizmadia et al., 2015). Selby and Woollard (2014) identify generalization as one of the most frequently occurring terms in CT literature. Moreover, they argued that generalization is used sparingly in the CT literature, but the idea of recognizing and reusing common parts of a solution or process is appropriate for inclusion in a definition of computational thinking. According to our analysis, generalization concept has 4 occurrences in CT definition category ([18] [28] [34] [74]), 1 occurrence in CT taxonomy category ([64]) and 3 occurrences in CT assessment category ([3] [27] [90]).

6. Discussion

The objective of this paper was to identify a basic set of domain independent dimensions of CT that researchers would agree upon. To date there is no consensus regarding the definitive or necessary components that would form the basic foundation of agreed CT concepts. According to the results presented in Table 1, abstraction and algorithms appear to be the prime prospects to build on consensus.

Results of our analysis in Table 1 are aligned with findings from previous works. Selby and Woolard (2014) conducted a research to identify a definition and a description of the core elements of CT. Based on the analysis, a consensus was found in the literature regarding abstraction concept and decomposition concept. Although consensus was not proposed for algorithmic design, the concept was found to be well defined across multiple disciplines and was included within CT definition. Araujo et al. (2016) found that algorithm and abstraction are the most commonly assessed CT skills. Lowe and Brophy (2017) argue that algorithms may be the most agreed CT concept across literature. Barr et al. (2011) report on consensus that emerged regarding the essential elements of CT during the meeting of a diverse group of educators with interest in CT from higher education, K-12 and industry. More than 82% of 697 respondents agreed or strongly agreed upon the essential elements of CT, where abstraction and algorithmic thinking were recognized as the essential concepts. Nevertheless, two literature review papers that were selected as the primary source of our study reveal comparably: Kalelioglu (2016) advocates that abstraction and algorithmic thinking are mostly used words in CT scope. Sondakh (2017) reports that abstraction and algorithms are the most commonly assessed CT concepts in higher education.

Results from Table 2 reveal that the main research focus is on defining and assessing CT skills and practices. This is aligned with the findings from Roman-Gonzalez et al. (2016) who claim that there is no consensus on formal definition of CT nor a consensus exists concerning how to effectively measure CT, although both aspects are in the main focus of recent research.

To line up with general, domain independent CT concepts, papers that were addressing domain specific CT skills or practices were excluded from analysis. This is particularly true for programming, while the majority of identified papers were using taxonomies where components were associated with programming specific concepts and artefacts. However, the results from our study can be applied to any domain, including programming.

To expand upon our study, teaching and assessing CT should include abstraction and algorithms. In this context, how should an instrument for assessing these practices look like? According to findings from our study, it depends on the domain that would be the subject of the research. If the aim would be to assess the algorithmic abilities of introductory computer science students or the algorithmic abilities of a software engineer, the instrument should be constructed as conceptual intervention within pseudocode or as Parson problem design, in both cases with an emphasis on algorithms. On the other hand, if the aim would be to assess the algorithmic abilities of a primary school aged child, the instrument should be constructed as CS Unplugged task focusing on sorting and searching.

7. Future work

Further research is needed towards the consensus from necessary to definitive set of domain independent components that define CT. A Delphi study appears to be a proper starting point for this task. The purpose of the Delphi study is to "obtain the most reliable consensus of opinion of a group of experts" using surveys and questionnaires (Linstone and Turoff, 2002, p.10). To our best knowledge, there were two Delphi studies conducted in the context of CT: Kolodziej (2017) conducted the Delphi study in order to formally define higher education curriculum, while Rich and Langton (2016) used Delphi process to formalize CT definition. Rich and Logan included only programming specific CT practices, therefore their result is not relevant in domain independent context.

The results of our study support previous findings concerning the complex, multidimensional nature of computational thinking. The proficiency of a child, a student or a professional within CT shows signs of being more complex than a simple aggregation of his or her proficiency within particular CT concept, such as abstraction or algorithms. In this context, further research is required to identify and evaluate significance of particular CT dimension within general problem solving proficiency of an individual. Several evaluation rubrics already exist in domain specific CT assessments, such as Dr. Scratch (Moreno-Leon et al., 2015), which is aimed to evaluate programming proficiency within the Scratch projects. Although the metric used by Dr. Scratch has been a subject of several evaluations Moreno-Leon et al. (2017) and Browning (2017), the utilized metrics treats each CT practice with equal significance. Thus, abstraction practice is equally important as the data representation practice. Moreover, the competencies level for "abstraction and decomposition" within Dr. Scratch should be evaluated with respect to theoretical computer science (Turner, 2018, p.149). From this perspective, competence level "definition of blocks" (Moreno-Leon et al., 2015, p.6) shows signs of being pure problem decomposition, or at least modularization and problem decomposition rather than abstraction and problem decomposition.

Last but not least, the results of our study also addressed challenges regarding teaching and assessing aspects of CT. There appears to be a consensus among the computer science researchers and educators concerning the multidimensional structure of CT. As a consequence, learning abstraction, algorithms or decomposition stimulates proficiency level of CT. Moreover, as advocated by Roman-Gonzalez et al. (2017, p.154) "CT assessment is an extremely relevant and urgent topic to address", researchers should focus on assessing particular dimensions of CT. Finally, our investigation on relevant CT concepts revealed the "multi face" structure of particular CT concepts: the treatment of abstraction and algorithms is not exactly the same within mathematics, computer science or within cognitive sciences. As CT is positioned as the umbrella of these "multi faced" concepts, further research is needed on possible "multi face" structure of computational thinking itself.

8. References

[1] Aho A.M. (2011) What is Computation? Computation and Computational Thinking. Ubiquity, an ACM publication. January, 2011, ACM New York, NY, USA, 1-8.

[2] Araujo A.L.S.O., Andrade W.L. and Guerrero D.D.S. A systematic mapping study on assessing computational thinking abilities. 2016 IEEE Frontiers in Education Conference (FIE), 1-9.

[3] Araujo A.L.S.O., Santos J.S., Andrade W.L., Guerrero D.D.S and Dagiene V. (2017) Exploring Computational Thinking Assessment in Introductory Programming Courses. 2017 IEEE Frontiers in Education Conference (FIE), 1-9.

[4] Barate A., Ludovico L.A. and Malchiodi D. (2017) Fostering Computational Thinking in Primary School through a LEGO-based Music Notation. International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France, 1334-1344.

[5] Barbara Sabitzer B., and Pasterk S. (2015) Modeling: A computer science concept for general education. 2015 IEEE Frontiers in Education Conference (FIE), 1-5.

[6] Barr D., Harrison J. and Conery L. (2011) Computational Thinking: A Digital Age. Learning & Leading with Technology, March/April 2011, 20-23.

[7] 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? Magazine ACM Inroads archive Volume 2 Issue 1, March 2011. ACM New York, NY, USA, 48-54.

[8] Basu S., Pollack R. and Roy M-R. (2006) Algorithms in Real Algebraic Geometry, Second edition. Springer-Verlag Berlin Heidelberg 2003, 2006, 281-291.

[9] Billionniere E. V. (2011) Assessing Cognitive Learning of Analytical Problem Solving. Doctoral dissertation, Arizona State University, December 2011.

[10] Bort H. and Brylow D. (2013) CS4Impact: Measuring Computational Thinking Concepts Present in CS4HS Participant Lesson Plans. Proceeding of the 44th ACM technical symposium on Computer science education, Denver, Colorado, USA — March 06 - 09, 2013, ACM New York, NY, USA, 427-432.

[11] Brennan K. and Resnick M. (2012) New frameworks for studying and assessing the development of computational thinking. Annual American Educational Research Association meeting 2012, Vancouver, BC, Canada, 1-25.

[12] Browning S.F. (2017) Using Dr. Scratch as a Formative Feedback Tool to Assess Computational Thinking. Master thesis, Brigham Young University 2017.

[13] Chen G., Shen J., Barth-Cohen L., Jiang S., Huang X. and Eltoukhy M. (2017) Assessing elementary students computational thinking in everyday reasoning and robotics programming. Computers and Education 109, 162-175.

[14] Colburn T. and Shute G. (2007) Abstraction in Computer Science. Minds & Machines (2007) 17,169-184.

[15] Committee for the Workshops on Computational Thinking, Computer Science and Telecommunications Board, Division on Engineering and Physical Sciences (2010) Report of a Workshop on the Scope and Nature of Computational Thinking (2010). The National Academies Press, Washington, D.C.

[16] Conery L.S., Stephenson C., Barr D., Barr V., Harrison J., James J. and Sykora C. (2011) Computational Thinking in K-12 Education teacher resources, Second edition.

[17] Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., and Woollard, J. (2015) Computational thinking: A guide for teachers. Computing at school.

[18] Curzon P., Dorling M., Ng T., Selby C. and Woollard J. (2014) Developing computational thinking in the classroom: a framework. Computing At School, 2014.

[19] Czarnecki K. (1998) Generative Programming - Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models. Doctoral dissertation, Technical University of Ilmenau, Germany, October 1998.

[20] Denning P.J. (2017) Remaining Trouble Spots with Computational Thinking. Magazine Communications of the ACM, Volume 60 Issue 6, June 2017. ACM New York, NY, USA, 33-39.

[21] diSessa A. (2018) Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education, Mathematical Thinking and Learning, 20:1, 3-31.

[22] Djambong T., Freiman V., Gauvin S., Paquet M. and Chiasson M. (2018) Measurement of Computational Thinking in K-12 Education: The Need for Innovative Practices. Digital Technologies: Sustainable Innovations for Improving Teaching and Learning. Springer International Publishing AG 2018, 193-222.

[23] Dong Y., Catete V., Jocius R., Lytle N., Barnes T., Albert J., Joshi D., Robinson R. and Andrews A. (2019) PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. Proceedings of the 50th ACM Technical Symposium on Computer Science Education, February 27–March 2, 2019, Minneapolis, MN, USA. 2019 Association for Computing Machinery, 906-912.

[24] Fetzer, J.H. (2001) Computers and Cognition: Why Minds Are Not Machines. Springer Science+Business Media Dordrecht, 101-130.

[25] Frorer P., Hazzan O. and Manes M. (1997) Revealing the faces of abstraction. International Journal of Computers for Mathematical Learning 2, 1997, 217-228.

[26] Futschek, G. (2006) Algorithmic Thinking: The Key for Understanding Computer Science. In Lecture Notes in Computer Science 4226, Springer, 159-168.

[27] Gouws L., Bradshaw K. and Wentworth P. (2013) First Year Student Performance in a Test for Computational Thinking. Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, ACM New York, NY, USA, 271-277.

[28] Grover S. and Pea R. (2013) Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher, Vol. 42 No. 1, 38-43.

[29] Habib M. and Paul C. (2010) A survey of the algorithmic aspects of modular decomposition. Computer Science Review, Volume 4, Issue 1, February 2010, 41-59.

[30] Hashemia N., Abua M.S., Kashefia H. and Rahimib K. (2013) Generalization in the Learning of Mathematics. 2nd International Seminar on Quality and Affordable Education (ISQAE 2013), 208-215.

[31] Hill R.K. (2015) What an Algorithm Is. Philos. Technol. (2016) 29. Springer Science+Business Media Dordrecht 2015, 35-59.

[32] Hsu T-C., Chang S-C. and Hung Y-T (2018) How to learn and how to teach computational thinking: Suggestions based on a review of the literature. Computers & Education 126 (2018), 296-310.

[33] Ingram-Goble A. (2013) Playable stories: making programming and 3d role-playing game design personally and socially relevant. Doctoral dissertation, Indiana University, October 2013.

[34] ISTE & CSTA. (2011) Operational Definition of Computational Thinking for K-12 Education.

[35] James Lockwood J. and Mooney A. (2018) Computational Thinking in Secondary Education: Where does it fit? A systematic literary review. International Journal of Computer Science Education in Schools, Jan 2018, Vol. 2, No. 1, 1-20.

[36] Kalelioglu F., Gulbahar Y. and Kukul V. (2016) A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, Vol. 4 (2016), No. 3, 583-596.

[37] Kamarudin K.M., Ridgwaya K. and Ismail N. (2016) Abstraction and Generalization in Conceptual Design Process: Involving Safety Principles in TRIZ-SDA Environment. Procedia CIRP 39 (2016), 16-21.

[38] Kao E. (2011) Exploring Computational Thinking at Google. The Voice of K-12 Computer Science Education and its Educators, Volume 7, Issue 2, May 2011, 6-7.

[39] Khine M. (2018) Computational Thinking in the STEM Disciplines, Foundations and Research Highlights. Springer International Publishing AG, part of Springer Nature 2018.

[40] Kitchenham B. and Charters S. (2007) Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. Engineering, Volume 45, Issue 4.

[41] Kleene, S.C. (1967). Mathematical Logic. New York: John Wiley and Sons.

[42] Knuth D.E. (1985) Algorithmic Thinking and Mathematical Thinking, The American Mathematical Monthly, 92:3, 170-181.

[43] Knuth D.E. (1997) The Art of Computer Programming: Volume 1: Fundamental Algorithms, Third Edition, Addison-Wesley, 1997.

[44] Kolodziej M. (2017) Computational Thinking In Curriculum For Higher Education. Doctoral dissertation, Pepperdine University, May 2017.

[45] Korkmaz O., Cakir R. and Ozden M. Y. (2017) A validity and reliability study of the computational thinking scales (CTS). Computers in Human Behavior 72 (2017), 558-569.

[46] Kramer J. (2007) Is abstraction the key to computing? Communications of the ACM April 2007/Vol. 50, No. 4, 37-42.

[47] Lesh R. and Harel G. (2003) Problem Solving, Modeling, and Local Conceptual Development, Mathematical Thinking and Learning, 5:2-3, 157-189.

[48] Linstone H.A. and Turoff M. (2002) The Delphi Method Techniques and Applications. https://web.njit.edu/~turoff/pubs/delphibook/delphibook.pdf, accessed 10.6.2019.

[49] Lishinski A., Yadav A., Enbody R. and Good J. (2016) The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1. Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, Tennessee, USA — March 02 - 05, 2016, ACM New York, NY, USA, 329-334.

[50] Lowe T. and Brophy S. (2017) An Operationalized Model for Defining Computational Thinking. 2017 IEEE Frontiers in Education Conference (FIE), 1-8.

[51] Marcelino M. J., Pessoa T., Vieira C., Salvador T. and Mendes A. J. (2018) Learning Computational Thinking and scratch at distance. Computers in Human Behavior 80 (2018), 470-477.

[52] Marr D.C. (2010) Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Mit Press 2010.

[53] Michaelson G. (2017) From Problems to Programs with Computational Thinking. http://www.macs.hw.ac.uk/~greg/Teaching%20Programming/From%20problems%20to%20programs%20with%20CT.pdf, accessed 10.6.2019.

[54] Moreno-Leon J., Carlos R.J., Harteveld C., Roman-Gonzalez M. and Robles G. (2017) On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts. Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, 2788-2795.

[55] Moreno-Leon J., Robles G. and Roman-Gonzalez M. (2015) Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. RED-Revista de Educacion a Distancia. Numero 46 15-Sep-2015, 1-23.

[56] National Research Council (2011) Report of a workshop of pedagogical aspects of computational thinking. Washington, D.C. National Academies Press.

[57] Nielsen N. (2018) Scientific Knowledge and Scientific Abstraction. https://medium.com/@jnnielsen/scientific-knowledge-and-scientific-abstraction-4adeb6589706, accessed 10.6.2019.

[58] Papert S. (1980) Mindstorms: children, computers, and powerful ideas. Basic Books, Inc USA.

[59] Parnas, D.L. (1972). On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM. 15(12), 1972, 1053-1058.

[60] Piaget J. (1950) The psychology of intelligence. London, UK: Routledge.

[61] Polya, G. (1957). How to Solve It: A New Aspect of Mathematical Method 2nd edn (Garden City, NJ, Doubleday).

[62] Prusinkiewicz P. (1998) In search of the right abstraction: The synergy between art, science, and information technology in the modeling of natural phenomena. C. Sommerer and L. Mignonneau (Eds.): Art @ Science. Springer, Wien, 1998, 60-68.

[63] Qualls J.A., Grant M.M. and Sherrell L.B. (2011) CS1 students' understanding of computational thinking concepts. Journal of Computing Sciences in Colleges, Volume 26 Issue 5, May 2011, Consortium for Computing Sciences in Colleges, USA, 62-71.

[64] Rad P., Roopae M., Beebe N., Shadaram M., Au Y. A. (2018) AI Thinking for Cloud Education Platform with Personalized Learning. Proceedings of the 51st Hawaii International Conference on System Sciences, 2018, 3-12.

[65] Rich P.J. and Langton M.B. (2016) Computational Thinking: Toward a Unifying Definition. J.M. Spector et al. (eds.), Competencies in Teaching, Learning and Educational Leadership in the Digital Age. Springer International Publishing Switzerland 2016, 229-242.

[66] Rodriguez B., Kennicutt S., Rader C. and Camp T. (2017) Assessing Computational Thinking in CS Unplugged Activities. Proceedings of the 2017 ACM SIGCSE Technical Symposium on

Computer Science Education, Seattle, Washington, USA — March 08 - 11, 2017, ACM New York, NY, USA, 501-506.

[67] Roman-Gonzalez M., Moreno-Leon J. and Robles G. (2017) Complementary Tools for Computational Thinking Assessment. Conference Proceedings of International Conference on Computational Thinking Education 2017. Hong Kong: The Education University of Hong Kong, 154-159.

[68] Roman-Gonzalez M., Perez-Gonzalez JC. and Jimenez-Fernandez C. (2016) Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior 72 (2017), 678-691.

[69] Romero M., Lepage A. and Lille B. (2017) Computational thinking development through creative programming in higher education. International Journal of Educational Technology in Higher Education (2017), 1-15.

[70] Rowe E., Cunningham K., Gasca S. (2017) Assessing Implicit Computational Thinking in Zoombinis Gameplay: Pizza Pass, Fleens & Bubblewonder Abyss. Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play, October 15–18, 2017, Amsterdam, Netherlands, 195-200.

[71] Sabitzer B., Demarle-Meusel H. and Jarnig M. (2018) Computational Thinking Through Modeling In Language Lessons. 2018 IEEE Global Engineering Education Conference (EDUCON), 1913-1918.

[72] Schwarz, C. V., Reiser, B. J., Davis, E. A., Kenyon, L., Acher, A., Fortus, D., Shwartz Y.,Hug B. and Krajcik J. (2009). Developing a learning progression for scientific modeling: Making scientific 147 modeling accessible and meaningful for learners. Journal of Research in Science Teaching, 46(6), 632-654.

[73] Seiter L. and Foreman B. (2013) Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. Proceedings of the ninth annual international ACM conference on International computing education research, ACM New York, NY, USA, 59-66.

[74] Selby C. and Woollard J. (2014) Refining an Understanding of Computational Thinking. University of Southampton Institutional Repository.

[75] Shaw I., Greene J.C., Mark M.M. (2013) The SAGE Handbook of Evaluation. SAGE Publications Ltd.

[76] Shute V.J., Sun C. and Asbell-Clarke J. (2017) Demystifying computational thinking. Educational Research Review 22 (2017), 142-158.

[77] Simon Sultana, S. (2016) Defining The Competencies, Programming Languages, And Assessments For An Introductory Computer Science Course. Doctoral dissertation, Old Dominon University, August, 2016.

[78] Sondakh D.E. (2017) Review of Computational Thinking Assessment in Higher Education. ResearchGate, https://www.researchgate.net/profile/Debby_Sondakh/publication/324984840_Review_of_Computational_Thinking_Assessment_in_Higher_Education/links/5af0378aa6fdcc85

[79] Stacewicz P. and Wlodarczyk A. (2010) Modeling in the Context of Computer Science - A Methodological Approach. Studies in Logic, Grammar And Rhetoric 20 (33) 2010.

[80] Stokes T. and Baer D. (1977) An implicit technology of generalization. Journal of Applied Behavior Analysis, 10, 349-367.

[81] Turner R. (2018) Computational Artifacts Towards a Philosophy of Computer Science. Springer-Verlag GmbH Germany, part of Springer Nature 2018.

[82] Walliman G. (2015) Genost: A System for Introductory Computer Science Education with a Focus on Computational Thinking. Doctoral dissertation, Arizona State University, May 2015.

[83] Webb H. C. and Rosson M. B. (2013) Using Scaffolded Examples to Teach Computational Thinking Concepts. Proceeding of the 44th ACM technical symposium on Computer science education, Denver, Colorado, USA - March 06 - 09, 2013. ACM New York, NY, USA, 95-100.

[84] Weintrop D., Beheshti E., Horn M., Orton K., Jona K., Trouille L. and Wilensky U. (2016) Defining Computational Thinking for Mathematics and Science Classrooms, Journal of Science Education and Technology, February 2016, Volume 25, Issue 1, 127-147.

[85] Werner L., Denner J. and Campe S. (2012) The fairy performance assessment: Measuring computational thinking in middle school. Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, North Carolina, USA — February 29 - March 03, 2012, ACM New York, NY, USA, 215-220.

[86] Whistler D. (2017) Abstraction and Utopia in Early German Idealism. Russian Journal of Philosophy & Humanities Volume 1 #2 2017, 3-22.

[87] Wing J.M. (2006) Computational thinking. Communications of the ACM - Self managed systems CACM, Volume 49 Issue 3, March 2006. ACM New York, NY, USA, 33-35.

[88] Wing J.M. (2008) Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society 2008, 3717-3725.

[89] Wing J.M. (2011) Research Notebook: Computational Thinking--What and Why? The Link. The magazine of the Carnegie Mellon University School of Computer Science. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why, accessed 10.6.2019.

[90] Yagci M. (2018) A valid and reliable tool for examining computational thinking skills. Education and Information Technologies, Springer Science+Business Media, LLC, part of Springer Nature 2018, 929-951.