Informatics in Education, 2020, Vol. 19, No. 1, 113–128
© 2020 *Vilnius University*
DOI: 10.15388/infedu.2020.06

A Model for Developing Computational Thinking Skills

Tauno PALTS, Margus PEDASTE
University of Tartu, Tartu, Estonia
e-mail: tauno.palts@ut.ee, margus.pedaste@ut.ee

Received: April 2019

Abstract. Computer science concepts have an important part in other subjects and thinking computationally is being recognized as an important skill for everyone, which leads to the increasing interest in developing computational thinking (CT) as early as at the comprehensive school level. Therefore, research is needed to have a common understanding of CT skills and develop a model to describe the dimensions of CT. Through a systematic literature review, using the EBSCO Discovery Service and the ACM Digital Library search, this paper presents an overview of the dimensions of CT defined in scientific papers. A model for developing CT skills in three stages is proposed: i) defining the problem, ii) solving the problem, and iii) analyzing the solution. Those three stages consist of ten CT skills: problem formulation, abstraction, problem reformulation, decomposition, data collection and analysis, algorithmic design, parallelization and iteration, automation, generalization, and evaluation.

Keywords: Computer science, STEM, problem solving, K-12, computational thinking.

1. Introduction

The 21st century has changed the type of skills, knowledge, and competences that are needed for success in the modern society. The reflection on computational thinking (CT) started from thinking about the way computer scientists think, but in the modern world it does not only involve computer scientists but has become a fundamental skill for everyone needing to find their way in the world of technology and solve problems effectively. Wing (2006) states that next to reading, writing, and arithmetic, CT should be added to everyone's analytical ability. Seymour Papert (1996) suggested a goal of introducing computational thinking by using a computer to solve problems in a way that allows people better to analyze and explain the problems, solutions and connections between them.

1.1. *Definitions and Dimensions of CT*

Thinking computationally includes various aspects, which could be taught already at the comprehensive school level. In 2006 Wing defined CT as “the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be carried out by an information-processing agent”. In that definition the information-processing agent is viewed as be a computer, a machine or just a human being. This definition is commonly referred to as a way to approach CT as a tool for solving problems using algorithms. The definition lacks more specific information about the dimensions of skills and aspects of developing CT, even though it still provides a frame of reference by depicting CT as a part of problem solving, including data representation, algorithmic thinking and a skill that can be developed either by using or not using technology. Qualls *et al.* (2011) connect the definition of CT even more directly to computer science, stating that it is using logic skills with computer science core concepts to solve problems. Grover and Pea (2013) describe CT as applying computer science tools and techniques for understanding natural and artificial processes and systems. Many other authors define CT as a skill, which is essential for programmers and computer scientists. For example, Anderson (2016) looks at CT as the way computer scientists think while approaching to solve the problems. Gouws *et al.* (2016) describe CT as a concept that extracts the thought processes involved in thinking like a computer scientist from concrete computer science practices and provides a more generalized understanding of how computer scientists approach problems.

In many cases, CT is mainly seen as an approach to solve problems. Soleimani *et al.* (2016) also suggest that CT is about planning and designing systems by using concepts of computer science. Getter and Yadav (2016) describe the CT including the concepts fundamental to computer science, along with the skills needed for abstraction. decomposition, pattern recognition and algorithmic thinking.

Selby and Woollard (2013) have extended Wing’s viewpoint by describing a five-dimensional model of CT skills being a thought process that reflects the ability to think:

- In abstractions, in order to understand the problem.
- In terms of decomposition, to divide the problem into smaller solvable problems.
- Algorithmically, to find the step by step solution for the problem.
- In terms of evaluations, to evaluate the effectiveness of the solution.
- In generalizations to be able to generalize the solution for wider range of problems.

Lee *et al.* (2011) emphasize that recognizing patterns as one core concept of CT defining CT as a set of thinking patterns. Roxcoe *et al.* (2014) view CT as a problem-solving technique that is a fundamental life skill.

CT skills have been recognized by computer science teaching organizations as part of the skillset that needs to be developed at various school levels. ISTE and CSTA (2011) offer a list of characteristics for describing CT that include formulating problems, organizing and analyzing data, using abstractions, automating solutions using algorithmic thinking with efficiency and generalizing solution for solving wide range of problems.

Large companies, like BBC and Google, have started programs to help develop children's CT skills. Google's CT course (2016) emphasizes manipulating data and confidence trait in the definition of CT. In this definition, CT is viewed as a process that includes logically ordering and analyzing data and creating solutions using algorithms, and dispositions to solve complex and open-ended problems.

CT is also claimed to be mainly a cognitive problem-solving process/ability, which can be developed in many ways, not only through computer programming (Chang *et al.*, 2017). CT is a skill that all everyone must learn to be effective at the workplace and to be ready for using the digital world (Vallance and Towndrow, 2016). Denning (2017) agrees that CT has evolved from being just the way that computer scientists think, to being useful in most other fields.

CT has been a topic of research for several years, with the specific aspects investigated including various age groups as well as tools for developing and instruments for measuring CT. As many authors have published various ways of defining and approaching CT, this leads us to the problem that not much attention has been dedicated to finding a common understanding of the dimensions of CT skills that would help us focus on developing and accessing CT skills.

1.2. *Problem*

CT has been described in several articles and reports, but these are not in line with each other and there is indeed missing a common understanding of the dimensions that should be in the focus while developing or assessing CT skills. There are several lists of CT skills describing CT, but no integrated model based on a common understanding of CT dimensions that can be used for developing CT skills.

In this study, using a systematic literature review method, we focus on finding a common understanding of the dimensions of CT skills that should be developed.

More specifically, two research questions were formulated:

(1) Which dimensions of CT skills can be identified in articles on developing CT?
(2) How can these dimensions from different articles be combined into a new theoretical model for developing CT?

2. Method

In order to identify the definitions and dimensions of CT and make a model for developing CT skills, a systematic literature review was conducted using the search engines EBSCO Discovery Service and the ACM Digital Library. The search was used to filter out the articles that included the search term "computational thinking" in the abstract.

2.1. *Search Procedure*

The search procedure started with specifying the search criteria to retrieve relevant academic articles (see Fig. 1). The search criteria set in the EBSCO Discovery Service search engine was as follows: (1) search term "computational thinking" in the abstract; (2) full text available; (3) peer reviewed, and (4) in English. As the ACM Digital Library search engine has slightly different search options, the following search criteria were set in the ACM Digital Library search engine: (1) search term "computational thinking" in the abstract, and (2) full text available.

The search was carried out on 1 January 2018 and returned 541 matches, including 228 in the ACM Digital Library and 313 in the EBSCO Discovery Service search results.

The next step was to filter out duplicate results (13), only 1–3 pages long texts (127) and those not written in the context of computer science education (32). Then, articles that did not include a clear list of CT skills (313) were excluded. Eventually, 9 articles were added based on references in selected articles. In total, 65 articles were included in qualitative analysis (see Fig. 1).

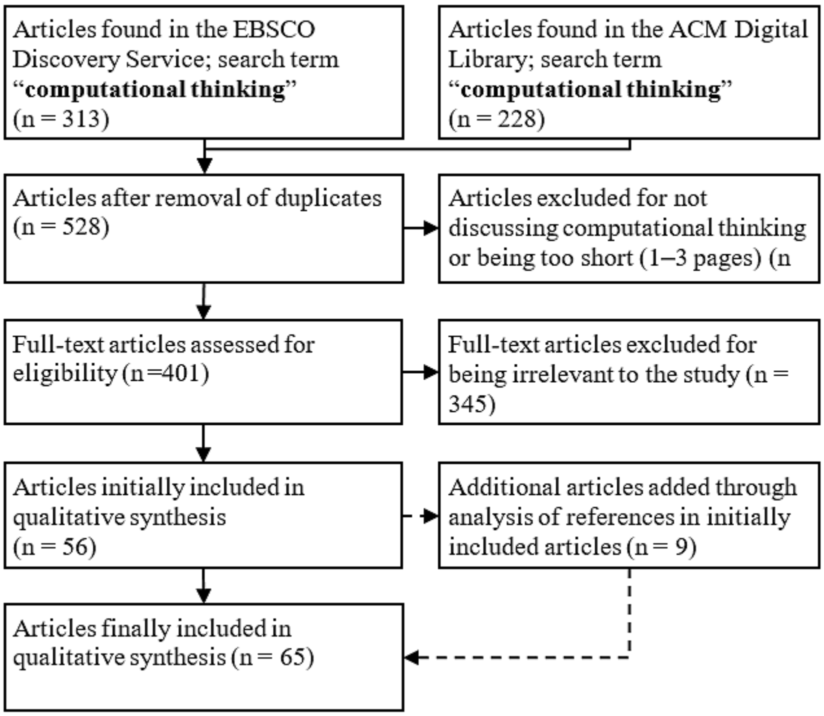
Articles found in the EBSCO Discovery Service; search term "**computational thinking**" (n = 313)
Articles found in the ACM Digital Library; search term "**computational thinking**" (n = 228)
Articles after removal of duplicates (n = 528)
Articles excluded for not discussing computational thinking or being too short (1–3 pages) (n
Full-text articles assessed for eligibility (n =401)
Full-text articles excluded for being irrelevant to the study (n = 345)
Articles initially included in qualitative synthesis (n = 56)
Additional articles added through analysis of references in initially included articles (n = 9)
Articles finally included in qualitative synthesis (n = 65)

Fig. 1. Stages of systematic CT literature analysis.

2.2. *Data Analysis*

Three steps of the systematic article review process for the 65 selected articles are as follows.

Firstly, in order to get an overview of the dimensions and CT skills described in the articles, an analytic framework was created, which included a reference name, model number and type, definition and dimensions of CT.

Secondly, a graph was formed based on the model type and number to reveal systematic descendancy of the CT models used in the articles (see Fig. 2). On this graph, main original articles that were mainly referred to, were highlighted and scientific articles with similar CT dimensions were connected with arrows.

Thirdly, as a result, a comparative analysis of the data from the 65 articles was conducted. The dimensions of CT skills were sequenced, and the descriptions of the dimensions by various authors the terms were combined in a table and later in a new model for developing CT skills. Finally, the model and the core concepts were described.

3. Findings and Discussion

Findings and discussion include sections about identifying the directions of CT, new model for developing CT skills and an example of using the new model.

3.1. *Identifying the Directions of CT*

In analyzing the 65 articles found in the study it appeared that they are often based on each other. Therefore, we started to categorize them based on the theoretical framework, definition and dimensions of CT used in each article. In addition, the year of publication was taken into account to characterize the descendancy of the articles. The results of this analysis of the content of the articles are presented on Fig. 2. Fig. 2 shows a graph of articles and the descendancy based on the six clusters of CT dimensions (framed in large boxes) that can be identified, originating from the following authors: Wing (2011), Barr and Stephenson (2011), CSTA and ISTE (2011), Brennan and Resnick (2012), Selby and Woollard (2013), and Moreno-León (2015).

The modern era of developing CT skills started with the article by Wing, describing CT as a fundamental skill for not only for computer scientists, but for everyone. The author analyzed the questions we might ask to solve an algorithmic problem by evaluating the difficulty and best solution for it. Wing emphasizes that the difficulty of the problem accounts for the power of the computing device (computer, machine or a human being) that will run the solution.

The first cluster of articles, deriving from **Wing** (2006), highlights the following characteristics of CT to be considered: abstraction, problem decomposition, problem reformulation, automation, and systematic testing.

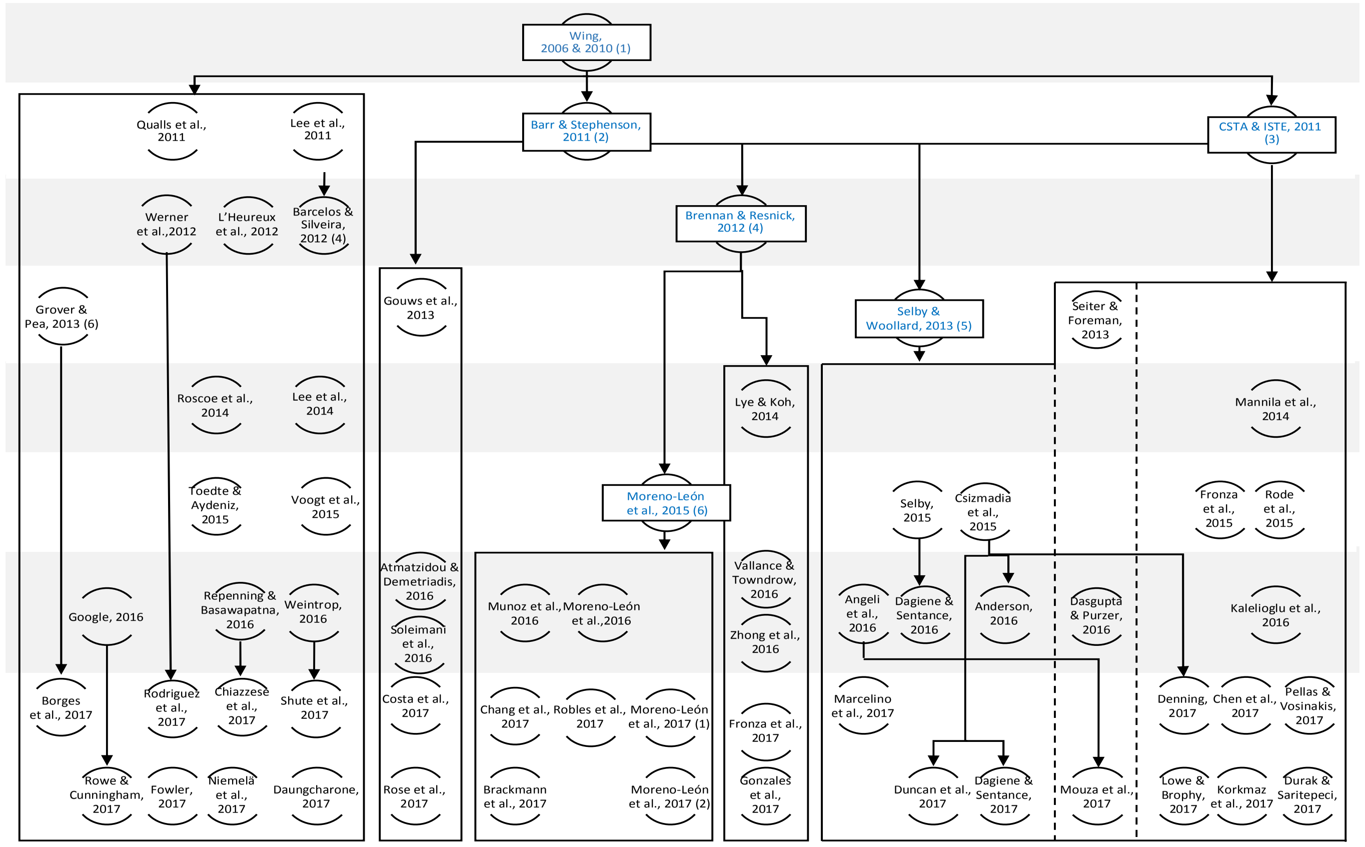

Fig. 2. Graph showing the relationships of the articles found in the systematic literature search. Gray shading indicates the year of publishing; the arrows show how previously published articles have been used as the theoretical rationale for the new articles; the blue font of the references with a surrounding square indicates key articles that have been further used in the synthesis of the current study.

Wing's viewpoint laid a foundation for a new wave of articles about the dimensions of CT skills. As seen on the Fig. 2, of all the selected articles, twenty-one present modifications of Wing's theory of characterizing CT.

The second cluster, deriving from Wing, started in 2011 when **Barr and Stephenson** (2011) created a comparison of CT concepts and capabilities in computer science, mathematics, science, language arts and social studies. Nine new common core CT concepts included three concepts of data manipulation: data collection, analysis and representation, and six problem solving concepts: decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation. The main difference compared to Wing's ideas was a greater focus on data manipulation and algorithms. In addition, parallelization and simulation were added as separate concepts of CT. This resulted in another wave of articles that use these dimension as a starting point for CT concepts (Gouws *et al.*, 2013, Soleimani *et al.*, 2016, Atmatzidou and Demetriadis 2016, Costa *et al.*, 2016 and Rose *et al.*, 2016).

The third cluster, deriving from Wing, appeared in 2011 when **CSTA and ISTE** (2011) used Wing's ideas to develop a list of six concepts for describing CT: formulating problems, organizing and analyzing data, abstractions, automation through algorithmic thinking, evaluation for efficiency and correctness, and generalizing. As we can see, the main focus of CT is considered as solving problems using algorithms. The main difference from Barr and Stephenson (2011) is that evaluation for efficiency and correctness and generalizing have been added as dimensions of CT. As international computer science teacher organizations have much influence on international teaching, several articles have used these concepts (Denning, 2017, Fronza *et al.*, 2015, Rode *et al.*, 2015, Kalelioglu *et al.*, 2016, Chen *et al.*, 2017, Pellas and Vosinakis, 2017, Korkmaz *et al.*, 2017, Durak and Saritepeci, 2017 and Lowe and Brophy, 2017).

The fourth cluster, deriving from Wing, emerged in 2011 when **Brennan and Resnick** (2012) described four practices to assess CT projects: abstracting and modularizing, reusing and remixing, being incremental and iterative, and testing and debugging. As Brennan and Resnick support the usage of Scratch as a tool for creating projects to develop CT, the focus is on project analysis. This type of analysis highlighted the new dimensions of iteration and reuse to be used in coding project analysis in several articles (Lye *et al.*, 2014, Vallance and Towndrow, 2016, Zhong *et al.*, 2016, Fronza *et al.*, 2017 and Román-González, 2017).

The fifth cluster, deriving from Barr and Stephenson (2011) and CSTA and ISTE (2011), was formed by the ideas of **Selby and Woollard** (2013) who identified in the literature the terms mostly associated with CT. They proposed a definition of CT, which includes the following terms: abstractions, decomposition, algorithmic thinking, generalization, and evaluation. Compared to Barr and Stephenson (2011), mostly data manipulation terms were left out for being either too broad, not-well defined or not considered a skill. Generalization and evaluation were added from CSTA and ISTE and those skills of CT have been used later by several authors (Anderson, 2016, Selby, 2015, Csizmadia *et al.*, 2015, Angeli *et al.*, 2016, Dagienė and Sentence, 2016, Marcelino *et al.*, 2016, Duncan *et al.*, 2017 and Dagienė *et al.*, 2017). As some of the dimensions of CSTA and ISTE (2011) are common with Selby and Woollard (2013), several authors have considered both ideas (Seiter and Foreman, 2013, Dasgupta and Purzer, 2016 and Mouza *et al.*, 2017).

The sixth cluster, deriving from Brennan and Resnick (2012), is based on connecting CT dimensions with automatic project analysis. In automatic assessment of Scratch projects, **Moreno-León** (2015) assesses the following CT aspects: abstraction in creating functions and clones, parallelism in starting several processes at the same time, logic in using logical operations, synchronization in sending messages, flow control in creating reasonable loops, user interactivity in using interaction, and data representation in using variables and lists in programs.

The main difference is that this approach has opened up algorithmic thinking as a demonstration of usage of parallelism, synchronization, logical thinking, and flow control. Furthermore, data manipulation has been emphasized by data representation and user interactivity. Although this categorization is influenced by automatic code assessment, it can be used in a number of block-based coding environments. Several authors have published articles analyzing only code used in educational programming environment (Chang *et al.*, 2017, Munoz *et al.*, 2016, Moreno-León *et al.*, 2016, Robles *et al.*, 2017, Moreno-León *et al.*, 2017a. Brackmann *et al.* 2017 and Moreno-León *et al.*, 2017b).

As the graph (see Fig. 2) shows, six main clusters of the dimensions of CT skills can be identified based on the various articles. Each cluster originates from certain authors, which leads us to the idea that, in order to form a unified model for developing CT skills, a collection of the CT skills from the original authors should be made.

Most of the articles have a common understanding of defining CT through the thinking process involved in solving algorithmic problems. Core concepts of CT are often described starting with defining the problem and ending with testing and evaluation. Solving problems is a cyclic process as solutions can be further on developed in the second cycle in the terms of practicality and efficiency. This leads to the idea of creating a model for developing CT skills in a cyclic manner including core concepts of CT divided into three main problem solving stages.

3.2. *New Model for Developing CT Skills*

Based on the six original articles describing the main skill clusters, CT skills can be grouped in three larger stages: defining the problem, solving the problem, and analyzing the solution (Table 1).

Comparison of these six models of CT skills gives us an opportunity to create a new model. This new model for developing CT skills (see Fig. 3) covers all of the main dimensions extracted from the articles in a three-staged problem-solving cycle, relying on CT as a way to solve problems algorithmically.

The stage of *defining the problem* includes all CT skills that are needed before starting to solve the problem. Firstly, from the Wing's (2006) definition of CT, problem solving starts with **formulating the problem**. Although several authors do not include it as a separate dimension, all of the articles describe it as part of the algorithmic problem-

Table 1
Categories of CT skills from six original articles

		Wing, 2006 [1]	**Barr, 2011 [18]**	**CSTA and ISTE, 2011 [12]**	**Brennan and Resnick, 2012 [19]**	**Selby and Woollard, 2013 [9]**	**Moreno-León et al., 2015 [20]**
Defining the problem	**Problem formulation**	Problem formulation	-	Formulating problems	-	-	-
	Abstraction	Abstraction	Abstraction	Abstractions	Abstracting and modularizing	Abstraction	Abstraction
	Problem reformulation	Problem reformulation	-	-	-	-	-
	Decomposition	Problem decomposition	Problem decomposition	-	-	Decomposition	Problem decomposition
Solving the problem	**Data collection and analysis**	-	Data collection, data analysis, data representation, simulation	Organizing and analyzing data,	Reusing and remixing	-	User interactivity, data representation
	Algorithmic design, parallelization and iteration, automation	Automation	Automation, algorithms and procedures, parallelization	Algorithmic thinking, automating solutions	Being incremental and iterative	Algorithmic design	Parallelism, logical thinking, synchronization, flow control
Analyzing the solution	**Generalization**	-	-	Generalizing	-	Generalization	-
	Testing and evaluation	Systematic testing	-	Identifying, analyzing, and implementing solutions	Testing and debugging	Evaluation	-

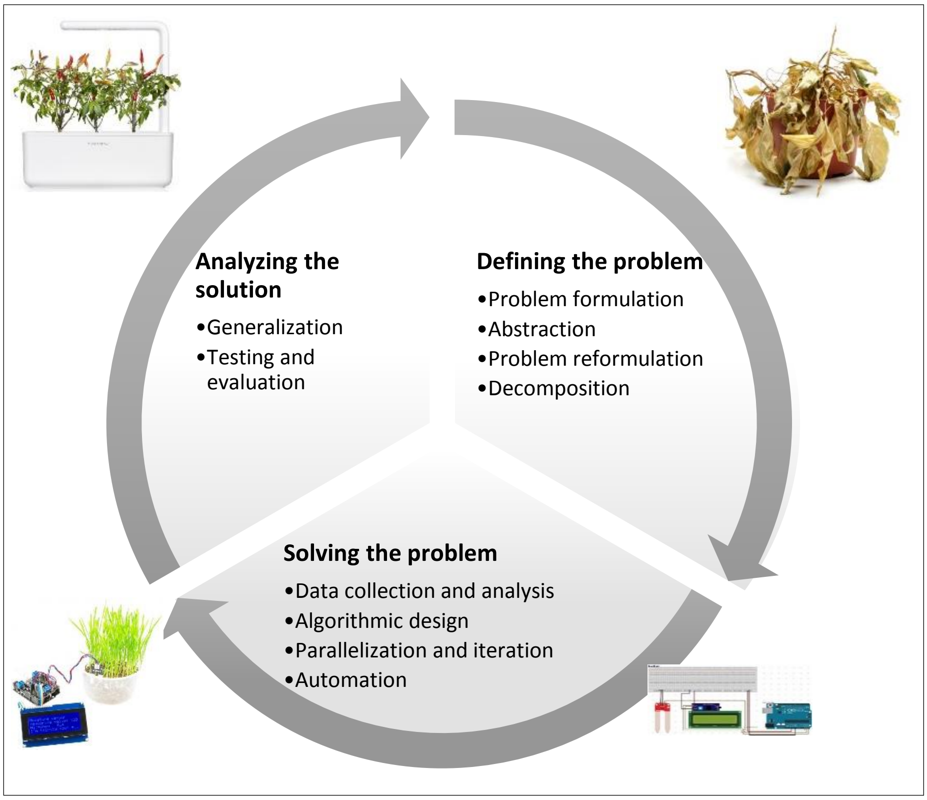

Fig. 3. A model for developing CT skills with illustrations from the project of measuring plant soil humidity.

solving process. As CT is a thought process used in solving problems according to algorithms, it is essential to understand and research the problem that needs to be solved. Secondly, all of the models include **abstraction**. When a problem is formulated, it is important to identify and extract relevant information to define the main idea(s). This stage includes modelling the core aspects of complex problems or systems. Therefore, the abstraction stage includes modularizing. Thirdly, **problem reformulation** can be used to reframe a problem into a solvable and familiar one. The fourth skill is **decomposition** of the problem. Usually, decomposing has been listed as the second dimension after the abstraction but, as two dimensions have been added, this step of breaking the problem down into manageable units now follows problem formulation.

The second stage of the model is *solving the problem*. This stage includes all CT skills involved in creating the solution for the problem. The precondition of solving the problem algorithmically is **collecting and analyzing data**. Another CT skill, **algorithmic design** (a series of ordered steps), is also a main core skill of CT. Algorithmic design is followed by the use of **parallelization and iteration**, which eventually leads to the **automation** of the process.

The third stage of CT skills is *analyzing the solution*. This includes **generalization**, which means transferring this problem-solving process to a wider range of problems. And the final CT skill is **evaluation and testing**, which means analyzing (assessing and recognizing) the processes and the outcomes in terms of efficiency and resource utilization. This also includes systematic testing and debugging, efficiency and performance constraints, error detection, etc.

When all of the CT stages have been completed, the solution can be improved after the evaluation and testing by formulating the problem again. This means repeating the three-staged application of CT skills until the user is satisfied with the result.

3.3. *Example of Using the New Model for Developing CT Skills*

Based on the six original articles describing the main skill clusters, CT skills can be grouped in three larger stages: defining the problem, solving the problem, and analyzing the solution (Table 1).

Comparison of these six models of CT skills gives us an opportunity to create a new model. This new model for developing CT skills (see Fig. 3) covers all of the main dimensions extracted from the articles in a three-staged problem-solving cycle, relying on CT as a way to solve problems algorithmically.

In order to get a better understanding of the new model for developing CT skills, this section provides an example of each step of the process, using a plant watering project as an illustration. Plants tend to lose vitality as a result of both excessive and insufficient watering (see Fig. 3). This example project is about solving the problem of watering the plants at home correctly.

In this case, an example of *formulating the problem* is: How to create a system that reminds us when to water the plant? For that we need to research the problem.

As an example of abstraction, we need to know if the plant needs watering. For that we need data on soil moisture (attribute for actual humidity as a percentage) and the minimum percentage of moisture needed by the specific plant (attribute for needed humidity as a percentage). Extra data is needed for turning on and off the alarm (attribute for alarm state as true or false). In addition to that we need code for measuring the soil humidity (method measure humidity) and code for turning the alarm on (methods turn alarm on) and off (method turn off the alarm). After the abstraction of understanding the problem, we need to decompose the problem into smaller solvable problems. Which tools can be used to measure humidity and set up an alarm? Which algorithms can turn the alarm on and off at certain humidity levels? An example of decomposing the plant watering system into solvable parts can be connecting the input (humidity sensor) and output (alarm) to the computer to read the humidity percentages and creating a program for reading the values of humidity and for turning on and off the alarm according to the humidity level. When the planning (see Fig. 3) is over, the second stage starts.

The second stage is *solving the problem*, which means creating suitable hardware and software solutions. For example, Arduino can be used to put together a device for measuring soil humidity (see Fig. 3). The next step would be to create the algorithm for measuring humidity and turning on the alarm when humidity is too low or too high.

This algorithm for humidity regulation can be used for a wide range of plants by just adjusting the variables for the minimum and maximum level of humidity that a specific plant needs. As humidity is measured, an if-else loop needs to be created with an algorithm that automatically turns on and off the alarm based on soil humidity.

The third stage is *analyzing the solution*. The CT skill of generalizing means that applicability of the solution can be expanded to measuring room humidity, temperature, light, etc. The skill of evaluating and testing the solution includes evaluation of the required frequency of taking humidity readings, the process of alarm transmission to notify the user and the accuracy of measurements and performance of the system.

After completing the three stages, the problem can be formulated again for improving the solution. For example, the user may not always be near the plant to water it or watering can be too labour-intensive for the owner. Would it be possible to add a water pump to improve the system? Can the design be more pleasing for the eye? Improving the solution can lead to actual working solutions (see Fig. 3).

When the prototype has passed through all three CT stages (see Fig. 3), the system can be improved in a cyclic manner after evaluation by formulating the problem again. The next step could be adding a watering system to the project. This means that the completed stages are followed again by three stages of CT skills until the user is satisfied with the result.

4. Conclusion

In this study, a new model for developing CT skills was designed based on an analysis of articles found through the EBSCO Discovery Search and the ACM Digital Library. Based on 65 articles, the study grouped the definitions and dimensions of CT in six clusters. The dimensions of CT from the six clusters were categorized into three sequenced stages in a new model. Usable in various subjects, the model includes the following stages:

- Defining the problem, which includes formulating the problem, abstraction, problem reformulation and decomposition.
- Solving the problem, which includes data collection and analysis, algorithmic design, parallelization and iteration and automation.
- Analyzing the solution, which includes generalization, testing and evaluation.

The new model also explains the stages and skills of CT with examples from project-based learning. The main new aspects of this model are that it can be used in a problem-solving process from start to finish by completing the sequential stages. Previous models have not listed CT skills in the order of occurrence in such a cyclic manner, with three major problem-solving stages following each other and each stage including CT skills that are developed at that stage. This model can be used by the teachers and students to develop various project ideas and instructional activities.

The limits of this model are that it is based only on the search term "computational thinking" and on the articles found by two search engines. The model does not include articles that do not present specific lists of CT skills. Furthermore, the identified clusters are based on a referring system and can change in the future, with new articles getting more references. Some aspects (for example abstraction) can occur in other dimensions, but the main occurrences are considered in the model.

Further research is needed to develop scenarios for training and assessing the development of CT skills at various school levels. Creation of tools for evaluating CT skills could give us more information about the relationships between the elements of the model. The model for developing CT skills is designed in such a way that it could be used in various subjects and at different school levels for developing CT skills.

The new model for developing CT skills is a theoretical one and we suggest creating scenarios of learning CT to test parts of it empirically in the future. When the prototype has passed through all three CT stages (see Fig. 3), the system can be improved after evaluation by formulating the problem again. For example, the next step could be adding a watering system to the project. This means that the completed stages are followed again by three stages of CT skills until the user is satisfied with the result.

References

Anderson, N. D. (2016). A call for computational thinking in undergraduate psychology. Psychology Learning & Teaching, 15(3), 226–234.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. Journal of Educational Technology & Society, 19(3), 47.

Atmatzidou, S., Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661–670.

Barcelos, T. S., Silveira, I. F. (2012, October). Teaching computational thinking in initial series an analysis of the confluence among mathematics and computer sciences in elementary education and its implications for higher education. In Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En (pp. 1–8). IEEE.

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.

Borges, K. S., de Menezes, C. S., da Cruz Fagundes, L. (2017, October). The use of computational thinking in digital fabrication projects a case study from the cognitive perspective. In 2017 IEEE Frontiers in Education Conference (FIE) (pp. 1–6). IEEE.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., Barone, D. (2017, November). Development of Computational Thinking Skills through Unplugged Activities in Primary School. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 65–72). ACM.

Brennan, K., Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (pp. 1–25).

Chang, C. K., Tsai, Y. T., Chin, Y. L. (2017, July). A Visualization Tool to Support Analyzing and Evaluating Scratch Projects. In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 498–502). IEEE.

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

Chiazzese, G., Fulantelli, G., Pipitone, V., Taibi, D. (2017, October). Promoting computational thinking and creativeness in primary school children. In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality (p. 6). ACM.

Costa, E. J. F., Campos, L. M. R. S., Guerrero, D. D. S. (2017, October). Computational thinking in mathematics education: A joint approach to encourage problem-solving ability. In 2017 IEEE Frontiers in Education Conference (FIE) (pp. 1–8). IEEE.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking-A guide for teachers.

CSTA. Operational Definition of Computational Thinking. 2011; `http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf`

Dagienė, V., & Sentance, S. (2016, October). It's computational thinking! Bebras tasks in the curriculum. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (pp. 28–39). Springer, Cham.

Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. Informatica, 28(1), 23–44.

Dasgupta, A., Purzer, S. (2016, October). No patterns in pattern recognition: A systematic literature review. In Frontiers in Education Conference (FIE), 2016 IEEE (pp. 1–3). IEEE.

Daungcharone, K. (2017, March). Enhancement the computational thinking skills via the simulation game. In Digital Arts, Media and Technology (ICDAMT), International Conference on (pp. 195–199). IEEE.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. Communications of the ACM, 60(6), 33–39.

Duncan, C., Bell, T., Atlas, J. (2017, January). What Do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum. In Proceedings of the Nineteenth Australasian Computing Education Conference (pp. 65–74). ACM.

Fowler, A. (2017, August). Engaging young learners in making games: an exploratory study. In Proceedings of the 12th International Conference on the Foundations of Digital Games (p. 65). ACM.

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

Fronza, I., El Ioini, N., & Corral, L. (2015). Students want to create apps: leveraging computational thinking to teach mobile software development. In Proceedings of the 16th annual conference on information technology education (pp. 21–26). ACM.

Fronza, I., Ioini, N. E., Corral, L. (2017). Teaching computational thinking using agile software engineering methods: a framework for middle schools. ACM Transactions on Computing Education (TOCE), 17(4), 19.

Google for Education. Exploring computational thinking. Retrieved December 13, 2016, from `http://www.google.com/edu/resources/programs/exploringcomputational-thinking/`

Gouws, L., Bradshaw, K., Wentworth, P. (2013, October). First year student performance in a test for computational thinking. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (pp. 271–277). ACM.

Gretter, S., Yadav, A. (2016). Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. TechTrends, 60(5), 510–516.

Grover, S., Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

Kalelioglu, F., Gülbahar, Y., Kukul, V. (2016). A framework for computational thinking based on a systematic research review. Baltic Journal of Modern Computing, 4(3), 583.

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

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32–37.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., and Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32–37.

Lee, T. Y., Mauriello, M. L., Ahn, J., Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. International Journal of Child-Computer Interaction, 2(1), 26–33.

L'Heureux, J., Boisvert, D., Cohen, R., Sanghera, K. (2012, October). IT problem solving: an implementation of computational thinking in information technology. In Proceedings of the 13th annual conference on Information technology education (pp. 183–188). ACM.

Lowe, T., Brophy, S. (2017). An operationalized model for defining computational thinking. In 2017 IEEE Frontiers in Education Conference (FIE) (pp. 1–8). IEEE.

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.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., Settle, A. (2014, June). Computational thinking in K-9 education. In Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference (pp. 1–29). ACM.

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., Mendes, A. J. (2017). Learning Computational Thinking and scratch at distance. Computers in Human Behavior.

Moreno-León, J., Robles, G., Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. RED. Revista de Educación a Distancia, (46), 1–23.

Moreno-León, J., Robles, G., Román-González, M. (2016, April). Comparing computational thinking development assessment scores with software complexity metrics. In Global Engineering Education Conference (EDUCON), 2016 IEEE (pp. 1040–1045). IEEE.

Moreno-León, J., Robles, G., Román-González, M. (2017a). Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. IEEE Transactions on Emerging Topics in Computing.

Moreno-León, J., Román-González, M., Harteveld, C., Robles, G. (2017b). On the automatic assessment of computational thinking skills: A comparison with human experts. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (pp. 2788–2795). ACM.

Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). Australasian Journal of Educational Technology, 33(3).

Munoz, R., Barcelos, T. S., Villarroel, R., Silveira, I. F. (2016, June). Game design workshop to develop computational thinking skills in teenagers with Autism Spectrum Disorders. In Information Systems and Technologies (CISTI), 2016 11th Iberian Conference on (pp. 1–4). IEEE.

Niemelä, P., Partanen, T., Harsu, M., Leppänen, L., Ihantola, P. (2017, November). Computational thinking as an emergent learning trajectory of mathematics. In Proceedings of the 17th Koli Calling Conference on Computing Education Research (pp. 70–79). ACM.

Papert, S. (1996). An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning, 1(1), 95–123.

Pellas, N., Vosinakis, S. (2017, April). How can a simulation game support the development of computational problem-solving strategies? In Global Engineering Education Conference (EDUCON), 2017 IEEE (pp. 1129–1136). IEEE.

Qualls, J. A., Grant, M. M., Sherrell, L. B. (2011). CS1 students' understanding of computational thinking concepts. Journal of Computing Sciences in Colleges, 26(5), 62–71.

Repenning, A., Basawapatna, A., Escherle, N. (2016, September). Computational thinking tools. In Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on (pp. 218–222). IEEE.

Robles, G., Moreno-León, J., Aivaloglou, E., Hermans, F. (2017, February). Software clones in Scratch projects: On the presence of copy-and-paste in Computational Thinking learning. In Software Clones (IWSC), 2017 IEEE 11th International Workshop on (pp. 1–7). IEEE.

Rode, J. A., Weibert, A., Marshall, A., Aal, K., von Rekowski, T., El Mimouni, H., Booker, J. (2015, September). From computational thinking to computational making. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (pp. 239–250). ACM.

Rodriguez, B., Kennicutt, S., Rader, C., Camp, T. (2017, March). Assessing Computational Thinking in CS Unplugged Activities. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 501–506). ACM.

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

Roscoe, J. F., Fearn, S., & Posey, E. (2014, October). Teaching computational thinking by playing games and building robots. In 2014 International Conference on Interactive Technologies and Games (pp. 9–12). IEEE.

Roscoe, J. F., Fearn, S., Posey, E. (2014, October). Teaching computational thinking by playing games and building robots. In Interactive Technologies and Games (iTAG), 2014 International Conference on (pp. 9–12). IEEE.

Rose, S., Habgood, J., Jay, T. (2017). An exploration of the role of visual programming tools in the development of young children's computational thinking. Electronic journal of e-learning, 15(4), 297–309.

Rowe, E., Asbell-Clarke, J., Gasca, S., Cunningham, K. (2017, August). Assessing implicit computational thinking in zoombinis gameplay. In Proceedings of the 12th International Conference on the Foundations of Digital Games (p. 45). ACM.
Seiter, L., Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. In Proceedings of the ninth annual international ACM conference on International computing education research (pp. 59–66). ACM.
Selby, C. (2015). Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In Proceedings of the Workshop in Primary and Secondary Computing Education (pp. 80–87). ACM.
Selby, C., Woollard, J. (2013). Computational thinking: the developing definition.
Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158.
Soleimani, A., Green, K. E., Herro, D., Walker, I. D. (2016, June). A Tangible, Story-Construction Process Employing Spatial, Computational-Thinking. In Proceedings of the The 15th International Conference on Interaction Design and Children (pp. 157–166). ACM.
Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. Journal of Educational Computing Research, 53(4), 562–590.
Toedte, R. J., Aydeniz, M. (2015). Computational thinking and impacts on K-12 science education. In Frontiers in Education Conference (FIE), 2015 IEEE (pp. 1–7). IEEE.
Vallance, M., and Towndrow, P. A. (2016). Pedagogic transformation, student-directed design and computational thinking. Pedagogies: An International Journal, 11(3), 218–234.
Werner, L., Denner, J., Campe, S., Kawamoto, D. C. (2012, February). The fairy performance assessment: measuring computational thinking in middle school. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 215–220). ACM.
Wing, J. (2011). Research notebook: Computational thinking—What and why. The Link Magazine, 20–23.
Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127–147.
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.

T. Palts is an Assistant of Informatics Didactics at the Institute of Computer Science of the University of Tartu and a PhD student in informatics. His primary research area relates to teaching computational thinking and programming at a beginner's level.

M. Pedaste is a Professor of Educational Technology at the Institute of Education of the University of Tartu where he is leading the Centre for Educational Technology. His research interests are in educational technology, science education, inquiry-based learning, technology-enhanced learning and instruction, learning analytics, and augmented reality.