

TAUNO PALTS

A Model for Assessing
Computational Thinking Skills



TAUNO PALTS

A Model for Assessing
Computational Thinking Skills



Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science Education on April 20, 2021 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof. Margus Pedaste
University of Tartu
Tartu, Estonia

Prof. Varmo Vene
University of Tartu
Tartu, Estonia

Opponents

Prof. Valentina Dagienė
Vilnius University
Vilnius, Lithuania

Prof. Erik Barendsen
Radboud University, Open University
Nijmegen, The Netherlands

The public defense will take place on June 8, 2021 at 14:15 in Narva mnt 18.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

Copyright © 2021 by Tauno Palts

ISSN 2613-5906

ISBN 978-9949-03-618-9 (print)

ISBN 978-9949-03-619-6 (PDF)

University of Tartu Press

<http://www.tyk.ee/>

To my family and friends

ABSTRACT

In the modernizing world, computer science is not only a separate discipline for scientists but has an essential role in many fields. Next to reading, writing, and arithmetic, thinking computationally has become a necessary skill for everyone. There is an increasing interest in developing computational thinking (CT) skills at various education levels – from kindergarten to university. Wing has 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 effectively carried out by an information-processing agent." [Win06]. Therefore, at the comprehensive school level, research is needed to have an understanding of the dimensions of CT skills and to develop a model to describe the dimensions of CT for assessing CT skills.

CT is described in several articles and reports. Still, these are not in line with each other, and there is missing a common understanding of the dimensions of the skills that should be in the focus while developing and assessing CT skills. In this doctoral study, through a systematic literature review, an overview of the dimensions of CT presented in scientific papers is given. A model for assessing 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.

The systematic development of CT skills needs an instrument for assessing CT skills at the basic school level. Tasks of the Bebras (Kobras) international challenge on informatics have been suggested as part of an instrument to assess CT skills. This doctoral study empirically tests which CT skills can be distinguished from the Bebras challenge results. Exploratory factor analysis was used to analyze the results from 7,100 participants of the Bebras challenge, and two main factors emerged, which can be characterized as algorithmic thinking and pattern recognition.

As an instrument is created for assessing CT skills at basic school, an interest remains, if and how this instrument for assessing CT skills can be used at the secondary school level. This doctoral study also presents a modified and empirically tested instrument with tasks for the secondary school level. 649 secondary school students were included in the study, and confirmatory factor analysis (CFA) was used to confirm that the modified instrument is suitable for assessing skills of algorithmic design and pattern recognition and can be used for setting directions for developing CT skills at the secondary school level.

Eventually, a modified model for assessing CT skills is presented, combining the theoretical and empirical results from the three main studies.

CONTENTS

List of original publications	11
1. Introduction	13
1.1. Research Problem	13
1.2. The Focus of the Research	15
2. Theoretical Background	17
2.1. Skills of CT	17
2.2. Assessing the Development of CT Skills	18
2.2.1. Assessing the Development of CT Using Gaming and Project Creation Activities	18
2.2.2. Assessing Development of CT Using Robotics and Tinkering Activities	21
2.2.3. Assessing Development of CT Using Unplugged Activities	22
3. Research Design and Methods	26
3.1. Research Design	26
3.2. Creating a Model for Assessing CT Skills	27
3.3. Methods for Developing Instrument to Assess CT in Basic School	28
3.4. Methods for Developing Instrument to Assess CT in Secondary School	29
4. Findings	31
4.1. A Model for Assessing CT Skills	31
4.1.1. Identifying the Directions of CT	31
4.2. A New Model for Assessing CT Skills	35
4.2.1. Defining the Problem	37
4.2.2. Solving the Problem	39
4.2.3. Analyzing the Solution	41
4.2.4. An Example of Using the New Model for Assessing CT Skills	43
4.3. Tasks for Assessing CT Skills at Basic School Level	45
4.3.1. Exploratory Factor Analysis to Explore the Factor Structure of Bebras Challenge Tasks to Assess CT	45
4.3.2. Description and an Example of Algorithmic Thinking Tasks	48
4.3.3. Description and an Example of Pattern Recognition Tasks .	49
4.4. Tasks for Assessing CT Skills at Secondary School Level	51
4.4.1. Pilot Study to Develop the Tasks to Assess CT Skills at Sec- ondary School Level	51
4.4.2. Confirmatory Factor Model to Confirm the Factor Structure of Tasks to Assess CT Skills	51
5. Discussion	54

6. Conclusions and Implications	57
6.1. Conclusions	57
6.2. Implications	58
6.3. Limitations	59
Bibliography	61
Appendix A. Tasks for Assessing CT at Basic School Level	69
A.1. Task 1. Geocaching	69
A.2. Task 2. Crane Operating	69
A.3. Task 3. Quick Beaver Code	70
A.4. Task 4. Mushrooms	70
A.5. Task 5. Biber Hotel	71
A.6. Task 6. Robot the Stairs	71
A.7. Task 7. Animation	72
A.8. Task 8. Fair Share	72
A.9. Task 9. Dream Dress	73
A.10. Task 10. Bracelet	73
A.11. Task 11. Cross Country	74
A.12. Task 12. Animal Competition	74
A.13. Task 13. Walnut Animals	75
A.14. Task 14. Button Game	75
A.15. Task 15. Pencils Alignment	76
Appendix B. Tasks for Assessing CT at Secondary School Level	77
B.1. Task 1. Crane operating	77
B.2. Task 2. Popularity	77
B.3. Task 3. Word Chains	77
B.4. Task 4. Geocaching	78
B.5. Task 5. Irrigation System	78
B.6. Task 6. Beaver Lunch	78
B.7. Task 7. Button Game	79
B.8. Task 8. Decorating Chocolate	79
B.9. Task 9. Pencils' Alignment	80
B.10. Task 10. Building a Chip	80
Sisukokkuvõte (Summary in Estonian)	81
Publications	83
Curriculum Vitae	128
Elulookirjeldus (Curriculum Vitae in Estonian)	129

LIST OF FIGURES

1. Example of a Scratch Project	20
2. LEGO Robots EV3 and NXT are suggested for developing CT skills	23
3. An Example of Bebras Challenge Task Suggested for Developing CT Skills	24
4. Research design	26
5. Stages of systematic CT literature analysis	27
6. Map of the clusters of the CT dimensions identified from the articles	32
7. A new model for assessing CT skills	37
8. A model for assessing CT skills with illustrations from the project measuring plants' soil humidity.	43
9. Solving the problem stage suggested for the model of assessing CT skills.	47
10. An example of algorithmic thinking task. Title: The Crane Operating	48
11. An example of an algorithmic thinking task. Title: The Quick Beaver Code	49
12. An example of a pattern recognition task. Title: The Animal Com- petition	50
13. An example of a pattern recognition task. Title: The Button Game	50
14. CFA model from the results of CT tasks (t1-t10) with the factor loadings of the two factors: algorithmic thinking (f1) and pattern recognition (f2).	52
15. A modified model for assessing CT skills	55

LIST OF TABLES

1. Categories of CT skills from six original articles	36
2. Factor loadings of two factors of the tasks and predicted original skill assessed.	46

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

- I **Tauno Palts** and Margus Pedaste. “A Model for Developing Computational Thinking Skills”. In: *Informatics in Education* 19.1 (2020), pp. 113–128. DOI: <http://dx.doi.org/10.15388/infedu.2020.06>.
- II **Tauno Palts** et al. “Tasks for Assessing Skills of Computational Thinking”. In: *Informatics in Education*. 10th annual International Conference of Education, Research and Innovation (2017), pp. 2750–2759. DOI: <http://dx.doi.org/10.21125/iceri.2017.0784>.
- III **Tauno Palts** and Margus Pedaste. “Tasks for Assessing Computational Thinking Skills at Secondary School Level”. In: (2019), pp. 216–226. DOI: http://dx.doi.org/10.1007/978-3-030-35343-8_23.

Other published work of the author

- IV Eerik Muuli et al. “Using image recognition to automatically assess programming tasks with graphical output”. In: *Education and Information Technologies* 25.6 (2020), pp. 5185–5203. DOI: <https://doi.org/10.1007/s10639-020-10218-z>.
- V Piret Luik et al. “Programming MOOCs – different learners and different motivation”. In: *International Journal of Lifelong Education* 39.3 (2020), pp. 305–318. DOI: <https://doi.org/10.1080/02601370.2020.1780329>.
- VI Piret Luik et al. “What motivates enrolment in programming MOOCs?” In: *British Journal of Educational Technology* 50.1 (2019), pp. 153–165. DOI: <https://doi.org/10.1111/bjet.12600>.
- VII Piret Luik et al. “Participants and Completers in Programming MOOCs”. In: *Education and Information Technologies* 24.6 (2019), pp. 3689–3706. DOI: <https://doi.org/10.1007/s10639-019-09954-8>.
- VIII Margus Pedaste et al. “Complex Problem Solving as a Construct of Inquiry, Computational Thinking and Mathematical Problem Solving”. In: 2161-377X (2019), pp. 227–231. DOI: <https://doi.org/10.1109/ICALT.2019.00071>.
- IX Marina Lepp et al. “Troubleshooters for Tasks of Introductory Programming MOOCs”. In: *The International Review of Research in Open and Distributed Learning* 19.4 (2018). DOI: <https://doi.org/10.19173/irrodl.v19i4.3639>.
- X Piret Luik et al. “Completion of Programming MOOC or Dropping Out: Are There Any Differences in Motivation”. In: (2018), pp. 329–337.

- XI Margus Pedaste et al. “What Happens to IT Education? The Case in Estonia with Some Recommendations for International Discussion”. In: *International Journal of Information and Education Technology* 7.3 (2017), p. 204.
- XII Marina Lepp et al. “Self-and Automated Assessment in Programming MOOCs”. In: (2016), pp. 72–85.
- XIII Marina Lepp et al. “MOOC in Programming: A Success Story”. In: (2017), pp. 138–147.
- XIV **Tauno Palts** and Margus Pedaste. “Tasks for Assessing Skills of Computational Thinking”. In: (2017), pp. 367–367.
- XV Külli Kori et al. “First-year Dropout in ICT Studies”. In: (2015), pp. 437–445.
- XVI **Tauno Palts** and Margus Pedaste. “Model of Learning Computational Thinking”. In: (2015), pp. 211–221.
- XVII Külli Kori et al. “What influences students to study information and communication technology”. In: *INTED2014 Proceedings* (2014), pp. 1477–1486.

1. INTRODUCTION

1.1. Research Problem

According to OECD [AGZ16], the European labor market lacks qualified ICT practitioners as jobs are lost and new jobs are created by automation. The report states a decreasing interest in STEM subjects science, technology, engineering, and mathematics. The problem is that in order to educate future citizens, educational research is needed to identify the forms of knowledge, skills, and competencies necessary for the advancement of society in the 21st century.

In computer science, questions arise, how to raise interest in and knowledge of learning the principles of computer science already at the comprehensive school level. At first, computational thinking (CT) was approached as a way computer scientists think, but recently it has been noted that it has become an essential way of thinking for everyone to solve problems effectively while finding their way in the world of technology.

In 1991 Seymour Papert mentioned the goal of introducing computational thinking – using a computer to solve problems so that it helps people analyze and explain the problems, solutions, and connections between them [Pap96]. Wing [Win06] states that CT skills are fundamental skills for everyone, not just for computer scientists, and belong to every child's analytical ability, just like reading, writing, and arithmetic. Wing started a new wave of developing CT as early as the comprehensive school level. Wing defined CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form which can be effectively carried out by an information-processing agent." [Win08]. The information processing agent can be a robot, a computer, a machine, or a human being. This definition of CT has been widely used and cited in further studies. Therefore, this Wing's definition of CT is used as a basis for this Ph.D. thesis.

Later, the term CT has had several interpretations and has evolved from being only a way programmers and computer scientists think to essential skills for every child. Some authors refer to CT as a way computer scientists think. For example, Anderson [And16] states that "Computational thinking is an approach to problem-solving typically employed by computer programmers.". Fronza et al. describe CT as a skill for everyone through in-depth computer science topics as being "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society." [FIC17]. Grover and Pea, 2013, describe CT as "the process of recognizing aspects of computation in the world that surrounds us and applying the tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes." [GP13].

Several authors describe CT as a way of thinking for computer scientists to tackle challenging problems. Gouws et al. state that "this concept 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." [GBW13]. Fowler defines it as "a method of using some of the concepts used in CS to systematically solve problems and process information." [Fow17]. Rose et al., 2017 add that a human being is an essential part of CT, describing CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." [RHJ17]. Shute et al., 2017, claim that CT occurs when students use computers to model their ideas and develop programs [SSA17]. These views on CT are based on the premise that everyone should be familiar with the basic concepts of computer science by developing CT.

Not all authors see CT as a skill for computer scientists. Vallance and Towndrow [VT16] claim that "CT is a skill that all pupils must learn if they are to be ready for the workplace and able to participate effectively in the digital world". Weintrop et al. [Wei+16] believe that CT "represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.". CT is needed for any workplace, but several authors include the problem-solving aspect of CT. For example, Lee et al. [Lee+11] look at CT in the context of problem-solving, claiming to use CT "to describe a set of thinking skills, habits, and approaches that are integral to solving complex problems using a computer and widely applicable in the information society". Voogt et al. [Voo+15] define CT as "a conceptual way to systematically, correctly, and efficiently process information and tasks to solve complex problems". Roscoe et al. [RFP14] state that CT "is a fundamental problem-solving technique that has applications beyond computing and is considered by many to be a fundamental life skill". This way, CT can be recognized as algorithmic thinking involved in solving problems.

The Computer Science Teacher Association (CSTA) and the International Society for Technology in Education (ISTE) describe CT as an ability to deal with complexity and open-ended problems [IC11]: "CT as a problem-solving process that includes (but is not limited to) the following characteristics: formulating problems for computational solution, logically organizing and analyzing data, abstractions including models and simulations, algorithmic thinking, evaluation for efficiency and correctness, generalizing and transferring to other domains".

Not all authors emphasize the role of problem-solving in CT. Selby and Woolard [SW13] define CT as "a cognitive or thought process that reflects the ability to think in abstractions, the ability to think in terms of decomposition, the ability to think algorithmically, the ability to think in terms of evaluations, and the ability to think in generalizations". Lee et al. look at CT as "a set of thinking patterns that includes understanding problems with appropriate representation, reasoning at multiple levels of abstraction, and developing automated solutions". These definitions concentrate on describing dimensions of CT, which mostly include describing problem-solving processes but do not name problem solving indirectly. Denning and Tedre [DT19] emphasize that CT varies as students progress taking a

constructivist approach by starting from writing down clear instructions and moving towards learning fundamental concepts found in programs. Although, they explain that CT is not a set of concepts for programming; it is a way of thinking that is honed through practice: the mental skills for designing computations to do jobs for us, and for explaining and interpreting the world as a complex of information processes. The authors identify six dimensions of CT: methods, machines, computing education, software engineering, computational science, and design.

Several reports have described CT, but these reports are not in line with each other. Various definitions of CT conflict with each other, for example, by stating that CT is essential for the way computer scientists think versus it is for everyone. Some of the definitions include the problem-solving aspect, and some do not. This leads to the idea of finding a common understanding of the dimensions of CT skills. Although several authors list CT skills, there is no integrated model based on a shared sense of CT dimensions that include CT skills for developing CT.

In addition to the common understanding of the definition and dimensions, a tool for assessing CT skills is needed. Such a tool would help us measure the development of CT skills at basic and secondary school levels in order to plan and develop various CT skills systematically.

1.2. The Focus of the Research

As there are many different definitions for CT, the aim of the study is to find a common understanding of CT and, more specifically, CT skills, to develop a model and an instrument for describing and assessing the dimensions of CT. Therefore, this Ph.D. study is divided into three studies: i) creating a model for developing CT skills, ii) creating an instrument to assess CT skills at the basic school level, and iii) creating an instrument to assess CT skills at the secondary school level.

The aim of the first study is as follows: Finding a common understanding of the dimensions of CT skills that should be developed at school. The following two research questions are posed for the first study:

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 assessing CT?

Research questions 1 and 2 are addressed in the original publication of Article I [TP20].

The aim of the second study is as follows: Finding and testing an instrument to assess CT skills in basic school. The following two research questions are posed for the second study:

3. Which instrument can be used to assess CT skills in basic school?

4. Which CT skills can be differentiated with an instrument used for assessing CT in basic school?

Research questions 3 and 4 are addressed in the original publication of Article II [Tau+17].

The aim of the third study is as follows: Finding and testing an instrument to assess CT skills in secondary school. The following research questions are posed for the third study:

5. Which instrument can be used to assess CT skills in secondary school?
6. Which CT skills can be differentiated with an instrument used for assessing CT in secondary school?

Research questions 5 and 6 are addressed in the original publication of Article III [TP19].

Visual representation of the research design including all three studies can be found in Figure 4.

2. THEORETICAL BACKGROUND

2.1. Skills of CT

International Society for Technology in Education (ISTE) has included CT as an across discipline for all the students, and CT is claimed to be a foundational skill for every student's ability to recognize opportunities to apply CT in their environment [IST]. As educators should develop a working knowledge of core components of CT, the question remains, which skills of CT should be developed already at the comprehensive school level?

Wing [Win08] described the skills of CT through problem-solving. Wing's definition includes formulating problems and their solutions so that the solutions can be carried out by a machine, a computer, or a human being. This definition has become the most used definition to describe a way to approach CT for solving problems using algorithms and is the base definition of CT in this thesis.

The Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) have collaborated to create a framework for preparing students to become computational thinkers who can use digital tools for solving future problems. They concluded that CT is a problem-solving process that includes (but is not limited to) the following six characteristics [IC11]:

- "Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data.
- Representing data through abstractions such as models and simulations.
- Automating solutions through algorithmic thinking.
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
- Generalizing and transferring this problem-solving process to a wide variety of problems."

The authors of computer science education articles have included various CT concepts in their models. For example, next to abstraction, problem decomposition, algorithms, and automation, Barr and Stephenson [BS11] have included the importance of data collection, analysis, and representation to the CT concepts. Also, parallelization and simulation should be considered to be the core CT concepts. Moreno-León et al. [MRR15] have created an automatic tool for assessing CT in educational programming language Scratch projects and have included the following skills of CT in their model: abstraction, parallelization, logic, synchronization, flow control, user interactivity, and data representation.

As CSTA and ISTE are leading computer science education organizations, most of the authors follow the previously mentioned list of the main ideas for developing CT [IC11], but that list is not that specific when it comes to teaching

the skills of CT. Selby and Woollard [SW13] have taken a more in-depth look at CSTA and ISTE suggestions. They have systematically identified CT skills incorporating thought processes that utilize abstraction, decomposition, algorithmic thinking, evaluation, and generalization. Logical thinking and problem solving are often mentioned as being part of CT, but Selby and Woollard [SW13] have excluded several previously mentioned terms due to being broad and not well defined. They claim that other aspects, such as system design, automation, computer science content, modeling, simulation, and visualization, are not skills but evidence of the use of skills. CT is an activity, often product-oriented, associated with, but not limited to, problem-solving. This leads us to five specific skills to be developed in comprehensive school curricula as skills of CT: abstraction, decomposition, algorithmic design, generalization, and evaluation.

The five skills of CT identified by Selby and Woollard [SW13] is a starting point for finding a common understanding of the CT skills in this thesis, but the question arises: Is this the list of CT skills that we should introduce at school, or are there any other skills that should be included in the model for developing CT?

2.2. Assessing the Development of CT Skills

The main focus of the thesis is on creating an instrument to assess CT skills. Instruments for assessing CT skills can vary depending on how we develop CT skills – instruments for assessing robotics research projects can differ from how we assess algorithmic tasks on paper. CT skills can be developed in various school subjects using multiple tools and methods.

Selby et al. [SDW14] have suggested six strands of CT: algorithms, programming and development, data and data representation, hardware and processing, communication, and networks and information technology. These strands can be used for creating scenarios for developing CT tasks but can overlap slightly. A pedagogical framework for CT [Kot+17] suggests dividing CT into four experiences: unplugged, tinkering, making, and remixing. Lee et al. have suggested teaching CT in youth practice in three domains: modeling and simulation, robotics and machines, and game design and development.

One common understanding of the categorization described above is by the tools used for developing and assessing CT. Looking at all of these categories, a common idea arises of dividing scenarios for developing and assessing the development of CT skills into three main categories: gaming and project creation, robotics and tinkering activities, and unplugged activities.

2.2.1. Assessing the Development of CT Using Gaming and Project Creation Activities

One of the options to develop CT skills is playing pre-made games for developing CT skills. Brackmann [Bra+17] has described tasks from the webpage code.org, designed to develop decomposition by breaking problems into smaller solvable

ones, e.g., planting a tree, washing hands, preparing breakfast, taking an elevator, tying shoes, etc. Algorithmic design, abstraction, and pattern recognition can be developed with games, including a map for finding the shortest route by using arrows and multipliers as commands. Also, a popular song was used as an example of how a song can turn into an algorithm. The game of Tetris was used to give instructions to the partner with the use of repetitions, moving every piece to the correct location. Eventually, a game was played, where several starting points had to be connected with ending points with various colors leaving no blank spaces. For assessing the CT skills, they used a test consisting of the following tasks: "Decomposition" activity (decomposition, algorithms), "Monica's Map" activity (pattern recognition and algorithms), "Elephants" activity (abstraction, pattern recognition, algorithms), "Tetris" activity (pattern recognition, algorithms), "Repetition Drawing" activity (decomposition, abstraction, pattern recognition, algorithms) and "Monica's Automata" (decomposition, abstraction, algorithms). These tasks seem to be engaging but assess several skills in each task and cannot be used for assessing one skill at a time.

Roman-Gonzalez [RPJ17] developed a CT test for assessing the skills of completion, debugging, and sequencing through nesting tasks The Maze and The Canvas. Those tasks involved finding the path through the labyrinth with correct sequencing arrows and drawing shapes on the canvas with valid commands. Results showed four factors emerging: verbal factor, spatial factor, reasoning factor, and numerical factor. Also, a problem-solving test was used to assess reasoning, spatial ability, and working memory. The problem-solving test had the most significant correlation with the CT test, but CT lacks information on specific CT skills.

Rose et al. [DS18] used Lightbot and Scratch jr tasks to develop CT skills. Firstly, matching shapes tasks were developed, where students had to select the odd one out of the shapes and select the next shape or the missing shape in a pattern. Answers were rotated, requiring the students to perform transformations like the thinking required in Lightbot. Two versions of Lightbot-like games with 15 levels were created – one that looked like Lightbot and another like Scratch jr. The CT skills of abstraction and generalization, algorithms and procedures, data collection, analysis and representation, decomposition, parallelism, debugging, testing and analysis, and control structures were developed with the tool. The authors assessed scores, steps, number of attempts, a level reached, and time for each student. This approach does not give much information about developing specific CT skills.

Brennan and Resnick [BR12] have suggested creating playful projects with the visual program language Scratch for developing CT skills. Completed project portfolios of students can be analyzed, artifact-based interviews conducted, and predesigned scenarios used for assessing the development of CT skills. Brennan and Resnick [BR12] claim that designing projects with Scratch develops CT concepts (sequences, loops, events, parallelism, conditionals, operators, and data),

practices (being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing), and perspectives (expressing, connecting and questioning). Resnick et al. [BR12] suggest three ways of assessing various aspects of CT in Scratch projects: project analysis, artifact-based interviews, and design scenarios. Fronza et al. [FIC17] used the same framework of analyzing Scratch projects for assessing animation creation, where the animal is moving towards the objects (see Figure 1), and eventually creating games, where the user clicks on objects to increase the score.

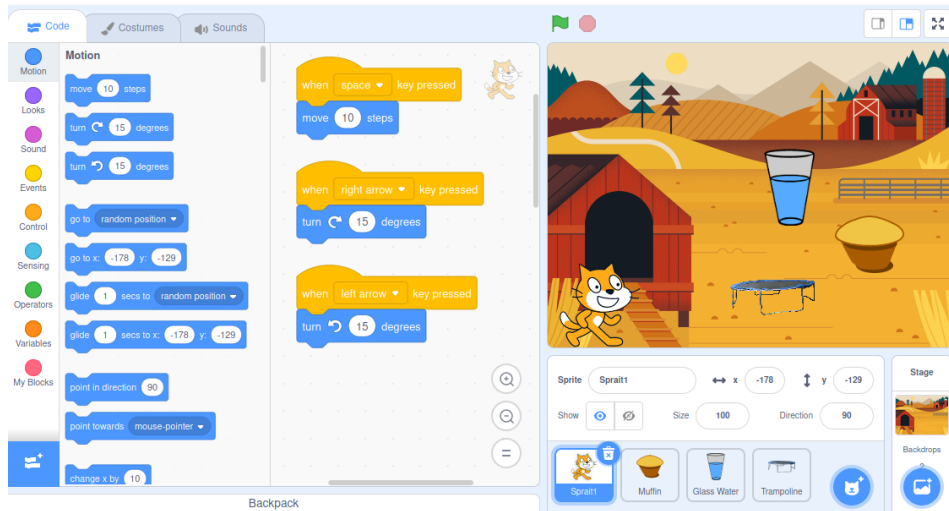


Figure 1. Example of a Scratch Project

Moreno-Leon et al. [MRR15] have developed a tool called Dr. Scratch for automated assessment of the Scratch project, which analyzes the usage of the following CT concepts: abstraction, parallelization, logic, synchronization, flow control, user interactivity, and data representation. Robles et al. [Rob+17] have emphasized that, in addition to creating original projects, the skill of reusing others' projects in Scratch can be useful as well as it develops the same CT skills. Chang et al. [CTC17] used the same tool for assessing the projects called Maze Starter and Slime Slayer and looked at the remixes that students composed. The results show that those remixes can develop other CT skills than original projects. Marcelino et al. [Mar+18] used Dr. Scratch for assessing results from a Scratch course with CT fundamentals, which involved constructing animations, interactive stories, and educational games. This tool seems to be strict but can assess Scratch projects that can be very different. Final projects included a poem that can be read and heard, a healthy food wheel game, where students have to put the different food in the right place of the wheel and gain some points that way, and an animal rights game with three different puzzles about the theme. Morelli et al. [Mor+11] suggest another way of developing CT by creating mobile apps with a

block-based environment, App Inventor, where usage of the same CT skills can be identified from the projects.

Similar to Scratch, a visual programming language environment Alice, has been used by Zhong et al. [Zho+16] for developing CT skills. They created six tasks for assessing sequences, loops, parallelism, modularization, testing and debugging, planning and designing, creativity and expressing, abstraction and modeling, testing and debugging, iteration and optimization, and reusing. Tasks included closed tasks where the student had to make a rabbit eat off a green cauliflower, a pair of semi-open tasks where students had to make the big rabbit jump to the front of red cauliflowers, and open tasks for designing a scenario to describe what has probably happened after the small rabbit became smaller and required filling out the creative design.

Another visual programming language environment, Kodu Gaming Lab, was used by Chiazzese et al. [Chi+17] to assess storyline (narrative), problem formulation (abstraction), solution expression (automation), execution, and evaluation (analysis). Tasks consisted of introducing visual programming and the Kodu environment by manipulating the physical tiles of the Kodu language. The next step was to design a virtual game scenario, define goals, choose rules, construct game actions, and create a scoring system. Children were guided by the autonomous construction of simple games, starting from the narrative description. Eventually, the acquired skills were assessed. There are other activities that can be included in this category, like a web page or mobile app creation.

2.2.2. Assessing Development of CT Using Robotics and Tinkering Activities

Robotics as a hands-on activity can be used for developing CT skills, too. Bers et al. [Ber+14] suggest the following TangibleK lessons for developing CT skills:

1. The engineering design process. Children solve toy people's transport problems by building non-robotic vehicles. For this purpose, the engineering design processes were implemented to plan, test, and improve their vehicles.
2. Robotics. This activity includes discussing the robots- what robots are and are not. Children explore robotic parts by designing and building their robots. The goal is to learn to connect robotic parts (wires and motors) to make a robot that moves. Children have an opportunity to build and design with the robotics materials freely and to create their programs beyond those that are outlined in each of the structured lessons.
3. Choosing and sequencing programming instructions. Children program robots to dance the "Hokey-Pokey" by choosing and sequencing relevant instructions. Those programming commands help to recall and apply the programming instructions.
4. Looping programs. Children use instructions to program robots to repeat

a movement forever. After that, they can program their robot's movement only a particular number of times to reach a fixed location.

5. Sensors. This activity includes using light sensors to program robots to turn on and off lights according to the darkness of the room. Children draw comparisons between robotic sensors and human senses.
6. Branching programs. Children are introduced to conditional control flow instructions, "If" and "If Not," which are used with sensors to make programs that include environmental conditions into the robot's behavior.

After the six activities, an interdisciplinary project can be made that invites children to apply the now-familiar powerful ideas to a different theme or context. The teacher decides on a theme drawn from other subjects studied during the year, and each child chooses a challenge within this theme, e.g., animal behaviors, vehicles that help the community, or "Who Am I?". Children created projects representing snakes that slither, recycling trucks that collect refuse, and sewing needles that travel back and forth through the fabric, among many others. The projects allow children to demonstrate the powerful ideas they learned over the six activities and apply them and continue learning about them in a new context. CT is assessed with a grading matrix, including aspects of problem-solving (hypothesis of the problem, working towards the solution, taking care of the workplace) and an interview asking questions about understanding the code and working towards the solution.

Atmatzidou and Demetriadis [AD16] suggest LEGO robotics (see Figure 2) for teaching CT skills. They studied 11 Lego Mindstorms NXT robotics lessons, which develop five CT skills: abstraction, generalization, algorithm, modularity, and decomposition. The following instruments were used to assess: profile questionnaire, two intermediate questionnaires, student opinion questionnaire, think-aloud protocol, interview, and observation.

There are other activities that can be included in this category, as the internet of things and smart clothing activities.

2.2.3. Assessing Development of CT Using Unplugged Activities

There are several ways to develop CT skills without the need for a computer, machine, or robot. Chiazese et al. [Chi+17] suggest paper graph activities for primary school pupils, as literature emphasizes the relevance of narrative strategies to stimulate learning processes. For this reason, the authors propose extending the range of CT skills by including a narrative stage as an introductory level. This stage includes writing the description of a story to be used in the virtual stage. The narrative stage is when students elaborate on the abstract formulation of a story (problem formulation) by extracting key information items from the narrative description of the story (storyline). It includes identifying characters, protagonists, antagonists, virtual stage elements, scoring rules, game goals, and characters' actions. After that stage, children can translate the key information items into pro-



Figure 2. LEGO Robots EV3 and NXT are suggested for developing CT skills

gramming instructions (solution expression). Finally, the children play the game and test (execution and evaluation) their code to detect errors. Various tools have been used to collect data: a pre-test and a post-test to detect differences in the attitude towards computer programming amongst students. The questionnaires have been delivered in the post-test, with the specific aims of measuring the reading, numeracy, and reasoning abilities and other measures of student performance. Unfortunately, the skills of CT were not explicitly tested.

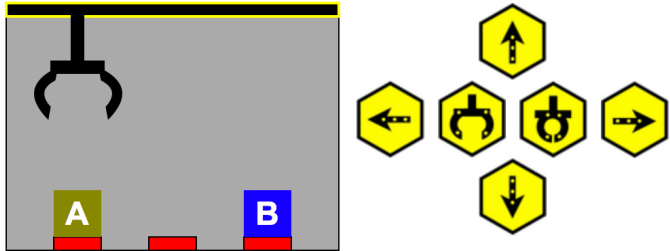
Borges et al. [BMC17] presented a challenge to develop a board game using, as much as possible, the tools available at the maker space (3D printer, laser cutter, vinyl cutter, and Arduino kits). Participants worked in groups of four people to create a digital portfolio using Google Documents and shared it with the author. Then they engaged in a 12-hour training for machine use (vinyl cutter, laser cutter, 3D printer, and basic electronics). After that, they had time to create and test a game. Initially, cognitive development was tested using Longleat's Test of Cognitive Development. This test aims to classify the subjects as being in either the concrete-operational or formal-operational stage. It evaluates the operational level of secondary school students in three areas: logic, combinations, and probabilities.

To assess specific skills of CT, Selby, Dorling, and Woollard [SDW14] created a matrix for assessing the five skills of CT: abstraction, decomposition, algorithmic thinking, generalization, and evaluation. This approach includes an individual assessment that might be time-consuming and has not been empirically tested in real-life situations. Dagiene and Sentance [DS16] and Dagiene et al. [DSS17] suggest Bebras challenge tasks for developing and assessing CT skills. Bebras is

an informatics challenge for pupils, which provides for many real-life-based tasks for developing CT. They have suggested that Bebras tasks can be used either as interesting starter tasks at the beginning of a lesson or in the formative assessment part of a lesson. They analyze 52 Bebras tasks that were chosen by Lithuania and the UK for all age groups. For each task, they identified the primary and most important CT skills being developed in that task. Twenty-two of the tasks involve some degree of algorithmic thinking in finding a solution. Eleven tasks involve the skill of evaluation, eight demonstrate abstraction, six decomposition, and five generalization. Tasks can demonstrate more than one CT skill, but in this instance, they have highlighted the most dominant one. Bebras tasks are short and designed to be solved within a few minutes. It can be challenging to generate tasks that demonstrate a lot of decomposition or evaluation in a short task. However, a key aspect of computer science at school is the design and execution of algorithms, which support the development of programming skills, so it may not be surprising that so many algorithmic thinking tasks (see Figure 3) make their way into the Bebras contest.

The crane in the port of Lodgedam has six different input commands:

left
right
up
down
grab
let go



Crate A is in the left position, crate B is in the position on the right.

Question:

Using the command buttons, swap the position of the two crates.

Figure 3. An Example of Bebras Challenge Task Suggested for Developing CT Skills

Dagiene and Sentence [DS16] suggest the following Bebras tasks (that can be found in UK Bebras "Computational Thinking Challenge "[B10+15]) for developing the following skills of CT:

- Abstraction is assessed in the tasks Beaver the Alchemist, Busy Beaver, Drawing Stars, Fried Egg, Geocaching, Popularity, Trains, and Walnut Animals.
- Decomposition is assessed in the tasks Animation, Fireworks, Pirate Hunters, Stack Computer, Quick Beaver Code, and Word Chains.

- Algorithmic thinking is assessed in the tasks Beaver Logs, Biber Hotel, Bowl Factory, Building a Chip, Button Game, Car Transportation, Chakhokhbili, Crane operating, Cross Country, Decorating Chocolate, Drawing Patterns, Dream Dress, Fair Share, Irrigation system, Left Turn!, Mushrooms, Pencils Alignment, Reaching the Target, Supper Power Family Theatre, Throw the Dice, and You Won't Find It.
- Evaluation is assessed in the tasks Animal Competition, Beaver Gates, Beaver Tutorials, Birds Bracelet, Birthday Balloons, Data Protection, Email Scam, Robot the Stairs, Setting the Table, and Turn the Cards.
- Generalization is assessed in the following tasks: Beaver Lunch, Kangaroo, Mobiles, RAID Array, and Spies.

Our study empirically tests the tasks by looking at the factors arising from the results of the Bebras challenge. Such tasks would help us choose specific tasks for developing and accessing specific skills of CT.

3. RESEARCH DESIGN AND METHODS

This chapter gives an overview of the research design and methods used in the doctoral study. Firstly, the research design, including flow charts of the studies and articles, is introduced. Secondly, methods for creating a model for assessing CT skills are described. Thirdly, methods for developing tools to assess CT skills at the basic and secondary school levels are presented. According to the International Standard Classification of Education, this study includes students from basic school classification ISCED 1 (11–12 years old, grade 5–6) and secondary school classification ISCED 3 (16–18 years old, grade 10–12).

3.1. Research Design

The doctoral study consists of three studies answering six research questions leading to a modified model for assessing CT skills (see Figure 4).

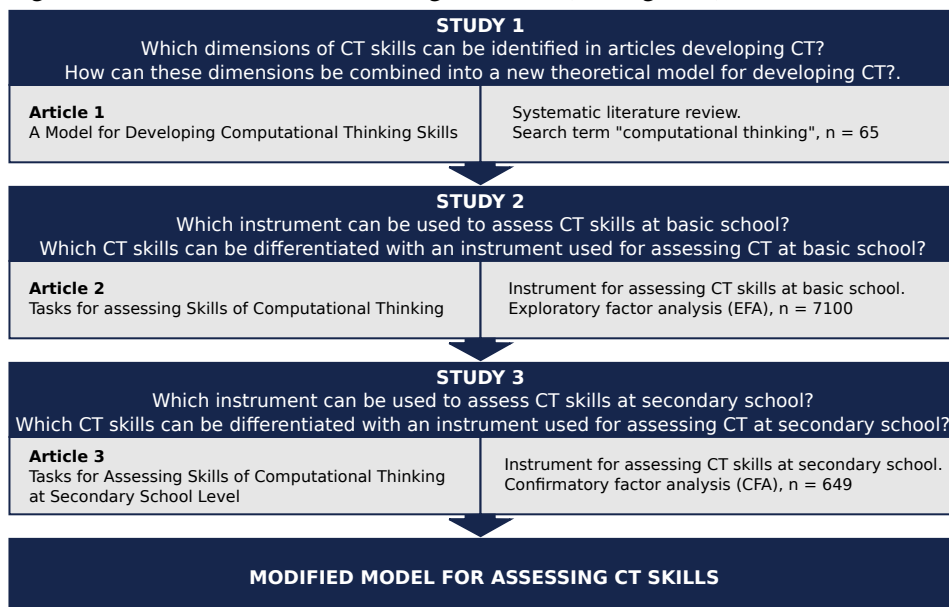


Figure 4. Research design

The goal of study 1 was to identify the dimensions of CT skills. A systematic literature review was conducted, and 65 articles were included to create a model for assessing CT skills. The goal of study 2 was to create an instrument for assessing CT skills in the basic school environment. Tasks from Bebras challenge [Blo+15] with the results from 7,100 students were analyzed, and a two-factorial model was created using exploratory factor analysis (EFA). The goal of study 3 was to adapt the tool created in study 2 for usage at the secondary school level. Results from 649 students were analyzed, and confirmatory factor analysis was

used to test the two-factorial model found at the basic school level. These three studies led to the result of creating a new model for assessing CT skills.

3.2. Creating a Model for Assessing CT Skills

The first study includes the identification of dimensions of CT to make a model for assessing CT skills. The PRISMA statement for reporting systematic reviews and meta-analyses of studies [Lib+09] was used as being one of the most used systematic review models in four stages: identification, screening, eligibility, and inclusion. This study used a systematic literature review method using EBSCO Discovery Service and ACM Digital Library search engines (see Figure 5). These search tools provided the main articles included in the introductory part of the thesis and are acknowledged search tools for scientific computer science education literature.

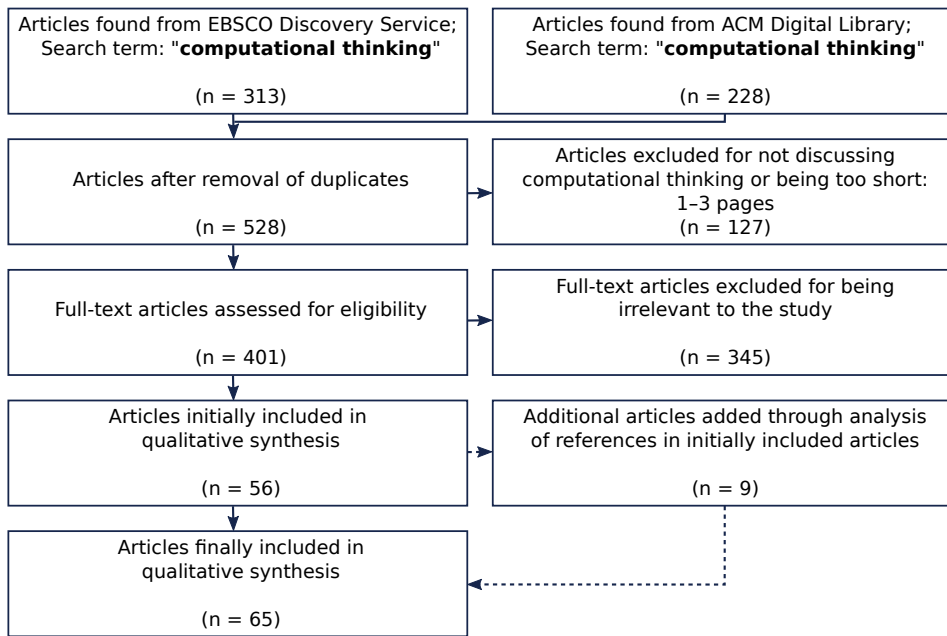


Figure 5. Stages of systematic CT literature analysis

The search procedure included the following steps:

1. **Step 1. Identification** Specifying the search of relevant articles, which was carried out on 1 January 2018. It returned 541 matches (228 from ACM Digital Library and 313 from EBSCO Discovery Service). As Wing in 2006 [Win06] started a wave of using the term "computational thinking," and the search was done in 2018, the publications' were from the period 2006 to 2018.

The following criteria were conducted in each search engine:

- (a) The EBSCO Discovery Service search engine: i) search term "computational thinking"; ii) full text available; iii) peer-reviewed, and iv) in English.
 - (b) The ACM Digital Library search engine: i) search term "computational thinking" in the abstract, and ii) full text available.
2. **Step 2. Screening** Filtering out relevant articles for the study. This step included filtering out:
- (a) duplicates (13);
 - (b) too short, only 1–3 pages long texts (127);
3. **Step 3. Eligibility.** In this stage, full articles were assessed for eligibility, and articles were excluded that:
- (a) did not include CT in the context of computer science education (32);
 - (b) had no exact listing of CT skills (313).
4. **Step 4. Inclusion.** During the qualitative synthesis of 56 articles, nine articles were added through analysis of the references. Added articles are connected with dashed lines in Figure 5).

During the filtering stage, nine articles were added based on the references in selected articles, making a total number of 65 articles used in the qualitative analysis (see Figure 5).

After filtering out 65 articles, a three-step systematic review process was conducted as follows. Firstly, to mark down the definitions and dimensions of CT skills described in the articles, an analytic framework was created, including a reference name, model type and number, definitions, and dimensions of CT skills. Secondly, descences of the models used in the articles were visualized by creating a cluster map (see Figure 6) and connecting the articles with arrows. This cluster map of the descences led to the mainly referred articles containing models of CT skills, which were highlighted. Thirdly, the CT skills from the mainly referred models were sequenced. Descriptions of the skills and dimensions by various authors were combined in a table (see Table 1) to create a new model for assessing CT skills. Finally, the model and the core CT skills were described with examples.

3.3. Methods for Developing Instrument to Assess CT in Basic School

The second study, after the systematic review of the articles, includes developing a tool for assessing CT skills at basic school. Basic school in Estonia includes grades from 1 to 9 (7–15 years old), and according to the International Standard Classification of Education, this study includes students from basic school classification ISCED 1 (11–12 years old, grade 5–6). Based on the five-dimensional model by Selby and Woollard [SW13], Bebras challenge tasks were suggested for

basic school by Dagiene and Sentance [DS16] to be used for assessing five skills of CT: abstraction, algorithmic design, decomposition, evaluation, and generalization. As Bebras is an award-winning international contest in informatics that has been running since 2004 in basic and secondary schools, with over 50 countries participating, the following tasks from UK Bebras "Computational Thinking Challenge" [Blo+15] were used for assessing mainly four CT skills out of five as no generalization tasks were suggested:

- **Abstraction:** Geocaching.
- **Decomposition:** Animation, Quick Beaver Code.
- **Algorithmic design:** Biber Hotel, Button Game, Crane Operating, Cross Country, Dream Dress, Fair Share, Mushrooms, Pencils Alignment.
- **Evaluation:** Animal Competition, Robot the Stairs.

Although the Bebras challenge is for six various age groups, in this study, the Benjamin age group (11–12 years of age) students from Lithuania ($n = 7,100$) were studied to determine which dimensions of CT skills can be identified from the results as independent factors. This study includes the results from all the Lithuanian students participating in the Bebras challenge preliminary round, including male and female students with various backgrounds.

For assessing CT skills at the basic school level, Bebras algorithmic problem-solving tasks were used with specific problem description and a question. Tasks have been translated by the organizers into Lithuanian and into English. Each task gave 1 point for correct answers, and -1 point for wrong answers. 0 points were given for no answer.

As the aim of this study was to understand if the tasks of the Bebras challenge assess the theoretical dimensions of CT skills as suggested, a confirmatory factor analysis (CFA) was used for finding out fit-indexes Tucker-Lewis (TLI), comparative fit (CFI), and root mean square error of approximation (RMSEA) for assessing CT skills as suggested. The Bebras challenge of 2015 used only tasks for assessing four out of the five dimensions of CT skills; therefore, CFA was performed for confirming the four-factor model. As CFA did not give the results theoretically expected, an exploratory factor analysis (EFA) was conducted to find out if results could be divided into categories that are not specific to the model with the five dimensions. EFA with principal axis factoring was used with the Oblimin rotation and Kaiser normalization to detect main factors. These factors are then interpreted by analyzing the tasks, as main CT skills in a problem-solving stage.

3.4. Methods for Developing Instrument to Assess CT in Secondary School

The third study presents a modification of the instrument from study 2 to assess CT skills at the secondary school level. According to the International Standard

Classification of Education, this study includes students from secondary school classification ISCED 3 (16-18 years old, grade 10-12).

For assessing CT skills of problem-solving at the secondary school level, a modified instrument from study 2 was created. In 2018 a pilot study for the third study and the main part of the third study was conducted.

A pilot study for the third study was conducted with 35 students from the 11th grade (16–17 years of age). Tasks with over 80% of the correct answers were replaced with the tasks suggested from the same dimension. The tasks for being suitable in the two-factorial model were Popularity, Word Chains, Irrigation System, Beaver Lunch, Decorating Chocolate, and Building a Chip [Blo+15]. A modified instrument for the secondary school includes ten tasks, five tasks from each dimension of CT skills – algorithmic thinking and pattern recognition. As the negative scores made statistically no difference, tasks were graded 1 point for correct answers and 0 points for incorrect answers. This instrument was uploaded as an online questionnaire, and the suggestion was to finish it within 40 minutes, but no strict time limit was set.

After the pilot study of the third study, a main part of the third study included students from the 10th grade of 17 secondary schools. All the schools were selected based on a voluntary basis, where the whole class, including male and female students with various backgrounds, had to complete the online test. 789 students started the test, but 118 results were removed for not finishing the test, and 25 were removed due to duplicates or students being unidentifiable. So the final study included a total number of 655 responses.

As the third study used a modified instrument from the second study, confirmatory factor analysis (CFA) was used to confirm the two dimensional CT skill structure found from the basic school study in the original publication Article I. CFA for validating the two-dimensional structure was done by determining the fit indices Tucker-Lewis (TLI), Comparative Fit (CFI), Root Mean Square Error of Approximation (RMSEA), and Weighted Root Mean Square Residual (WRMR).

As two factors were predicted, a two-factor model was created with five tasks in each factor. The statistical program Mplus (Version 7) [BG11]) was used for CFA. In order to evaluate the models, we adopted criteria for fit indexes that had been proposed by Mplus [MM98], which are as follows: RMSEA: close fit: $\leq .05$, reasonable fit: $.05-.08$, poor fit: $\geq .10$; CFI: $\geq .95$; TLI: $\geq .95$).

4. FINDINGS

4.1. A Model for Assessing CT Skills

This chapter describes the process of identifying the directions of CT with a new proposed model for assessing CT skills. CT skills identified in this model are used for creating tasks for assessing CT skills at basic school and secondary school levels.

4.1.1. Identifying the Directions of CT

CT has been a matter of discussion for several years, and opinions on the dimensions of CT have evolved accordingly. This study concentrates on that issue by including 65 articles for identifying the skills of CT. It appeared that those articles are often referring to each other. Therefore, for a better understanding of the relations between various articles about CT skills, articles were categorized based on the theoretical framework, definition, and skills of CT. The year of publication helps to create the dependencies of the articles as usually, newer articles are based on the previous studies. As a result of the article content analysis, a cluster map was created (see Figure 6), which demonstrates the descendancies of the articles.

Large colored framed borders on Figure 6 mark six bigger clusters of identified CT skills from the original publication Article I that originated from the following authors:

1. Wing (2006) [Win06],
2. Barr and Stephenson (2011) [BS11],
3. CSTA and ISTE (2011) [IC11],
4. Brennan and Resnick (2012) [BR12],
5. Selby and Woollard (2013) [SW13], and
6. Moreno-León (2015) [MRR15].

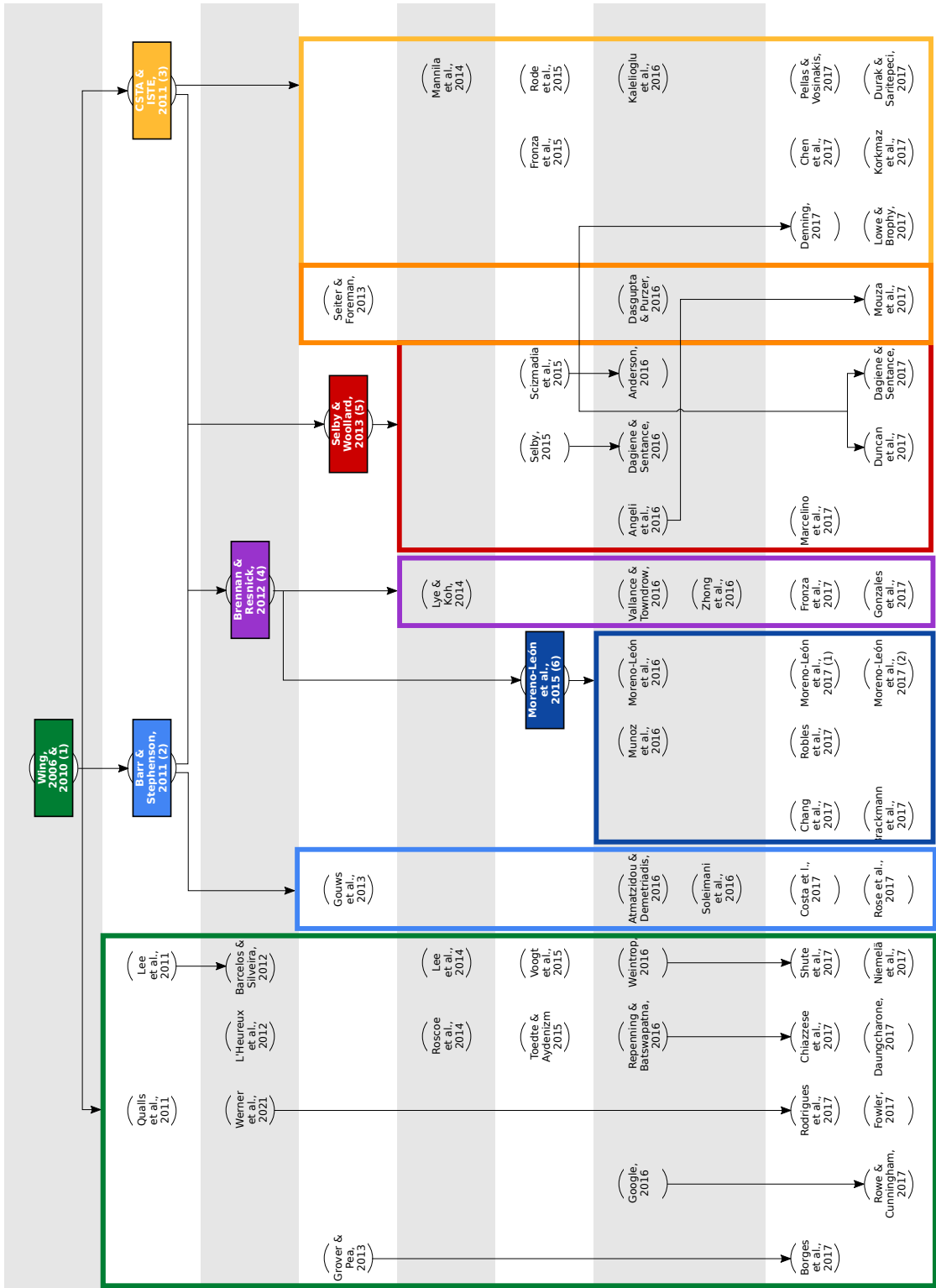


Figure 6. Map of the clusters of the CT dimensions identified from the articles

As the cluster map shows, the modern era of developing CT skills started with Wing in 2006 [Win06], suggesting CT for describing fundamental skills for everyone, not only for computer scientists. CT skills are connected with solving algorithmic problems with a computer, machine, or human being. Solving algorithmic problems includes evaluating the difficulty and best solutions for solving the accounting problems for the power of the computing device that runs the system.

The first cluster (on the left side of the cluster map on Figure reffig:clusterm, with a green border) includes twenty-one articles that derive from Wing's theory [Win06] highlighting the following characteristics of CT: abstraction, problem decomposition, problem reformulation, automation, and testing.

The second cluster, deriving from Wing [Win06], starts from Barr and Stephenson (2011) [BS11] (light blue border on Figure 6). They compared nine CT concepts and capabilities in computer science, mathematics, science, language arts, and social studies: data collection, analysis and representation, decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation. Compared to Wing [Win06], these concepts have a more significant focus on data manipulation (collection, analysis, and representation) and algorithms. Parallelization and simulation were added as separate core CT concepts. Deriving from this article, the second cluster of articles uses these CT concepts (Gouws et al., 2013 [GBW13], Soleimani et al., 2016 [Sol+16], Atmatzidou and Demetriadis 2016 [AD16], Costa et al., 2016 [CCG17] and Rose et al., 2017 [RHJ17]).

The third cluster, deriving from Wing [Win06], starts from the six concepts presented by CSTA and ISTE (2011) [IC11] (dark blue border on Figure 6), which includes formulating problems, organizing and analyzing data, abstractions, automation (algorithmic thinking), evaluation for efficiency and correctness, and generalizing. The main focus of these concepts is on solving problems using algorithms. These concepts are different from Barr and Stephenson (2011) [BS11] as generalization and evaluation for efficiency and correctness were included as CT skills. As computer science teacher organizations (ISTE and CSTA) have an international influence on teaching, nine articles have used those CT concepts (Denning, 2017 [Den17], Fronza et al., 2015 [FIC17], Rode et al., 2015 [Rod+15], Kalelioglu et al., 2016 [KGK16], Chen et al., 2017 [Che+17], Pellas and Vosinakis, 2017 [PV17], Korkmaz et al., 2017 [KÇÖ17], Durak and Saritepeci, 2018 [DS18] and Lowe and Brophy, 2017 [LB17]).

The fourth cluster, deriving from Wing [Win06], comes from Brennan and Resnick (2012) [BR12] (pink border on Figure 6), studied Scratch projects and described four practices of CT: abstracting and modularizing, reusing and remixing, being incremental and iterative, and testing and debugging. As block-based coding environments like Scratch have been suggested for developing CT skills, the fourth cluster includes five articles covering those practices of CT (Lye et al., 2014 [LK14], Vallance and Towndrow, 2016 [VT16], Zhong et al., 2016 [Zho+16], Fronza et al., 2017 [FIC17] and Román-González, 2017 [RPJ17]).

The fifth cluster includes articles deriving from Selby and Woollard (2013) [BR12] (red border in Figure 6). They synthesized the CT dimensions from Barr and Stephenson (2011) [BS11] and CSTA, and ISTE (2011) [IC11]. Selby and Woollard [BR12] concluded that CT skills involve abstractions, decomposition, algorithmic thinking, generalization, and evaluation. Compared to Barr and Stephenson (2011) [BS11], data manipulation was left out for being too broad, not-well defined, or not considered a skill. The CT skill of generalization was added from the dimensions of CSTA and ISTE [IC11]. Afterwards, those five skills of CT have been recognized as being the main dimensions of CT by several authors (Anderson, 2016 [And16], Selby, 2015 [Sel15], Csizmadia et al., 2015 [Csi+15], Angeli et al., 2016 [Ang+16], Dagienė and Sentence, 2016 [DS16], Marcelino et al., 2018 [Mar+18], Duncan et al., 2017 [DBA17] and Dagienė et al., 2017 [DSS17]. As some of the dimensions of CT from CSTA and ISTE (2011) [IC11] are common with Selby and Woollard (2013) [SW13] (orange border on Figure 6), several authors have included both ideas (Seiter and Foreman, 2013 [SF13], Dasgupta and Purzer, 2016 [DP16] and Mouza et al., 2017 [Mou+17]).

The sixth cluster evolves from the article by Brennan and Resnick (2012) [BR12] (yellow border on Figure 6), which connects it with an analysis of automatic visual programming language projects. Moreno-León (2015) [MRR15] defined eight aspects of CT that can be assessed in Scratch projects: abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. The main impact is opening up algorithmic thinking by looking at it to demonstrate the usage of parallelism, synchronization, logical thinking, and flow control. As articles in the previous dimension have excluded data manipulation, Moreno-León (2015) [MRR15] has included it due to being emphasized by data representation and user interactivity. These dimensions have been used later on in several studies (Chang et al., 2017 [CTC17], Munoz et al., 2016 [Mun+16], Moreno-León et al., 2016 [MRR16], Robles et al., 2017 [Rob+17], Moreno-León et al., 2017a [MRR17], Brackmann et al. 2017 [Bra+17] and Moreno-León et al., 2017b [Mor+17]) that analyze projects created in educational programming environments.

As the cluster map (see Figure 6) shows, six main clusters of the dimensions of CT skills can be identified based on the articles. Each cluster derives from specific authors, but no real consensus about the dimensions has appeared. Each cluster originating from distinct authors leads us to the idea of collecting the CT skills from each original author to form a unified model for assessing CT skills. The common understanding from the articles is that CT is the thinking process involved in solving algorithmic problems. Wing [Win06] states that problem-solving starts with defining a problem and ends with testing and evaluation.

The next section describes the process of creating a model for assessing CT skills based on the composed cluster model.

4.2. A New Model for Assessing CT Skills

Based on 65 included articles, the first study of the thesis grouped the definitions and dimensions of CT into six clusters identified from the original articles found through the EBSCO Discovery Search and the ACM Digital Library. To find a common understanding of the relations between the aforementioned six studies, a proposal of using three stages of the CT process from Repenning et al. (2017) [RBE17] was used. Repenning [RBE17] identified three stages of the CT process that a CT tool must elicit:

1. Problem formulation (abstraction), where the problem is conceptualized verbally.
2. Solution expression (automation), where the solution is expressed in a non-ambiguous way so that the computer can carry it out.
3. Execution and evaluation (analysis), where the solution is executed by the computer in ways that show the direct consequences of one's thinking.

Finding a consensus can be difficult, but based on the descriptions of the CT skills, the first study of the thesis suggests grouping all the CT skills deriving from the six original articles into three larger algorithmic problem-solving stages. In this thesis, Table 1 visualizes how all the CT skills deriving from the six original articles can be grouped into three larger algorithmic problem-solving stages: defining the problem, solving the problem, and analyzing the solution (see Table 1). Defining the problem includes skills problem formulation, abstraction, problem reformulation, and decomposition. Solving the problem includes skills in data collection and analysis, algorithmic design, automation, parallelization, and iteration. Analyzing the solution includes generalization, testing, and evaluation.

Table 1. Categories of CT skills from six original articles

	Wing, 2006	Barr & Stephenson, 2011	CSTA & ISTE, 2011	Brennan & Resnick, 2012	Selby & Woollard, 2013	Moreno-Leon et al., 2015
Defining the problem	Problem formulation Abstraction	Abstraction	Formulating problems Abstraction	- Abstracting and modularizing	- Abstraction	- Abstraction
problem reformulation and decomposition.	Problem reformulation Problem decomposition	- - Problem decomposition	- - - -	- - - -	- - Decomposition	- - Problem decomposition
Solving the	-	Data collection, data analysis, data representation, simulation	Organizing and analyzing data	Reusing and remixing	-	User interactivity, data representation
problem	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	- Systematic testing	- - -	Generalizing Identifying, and analyzing, and implementing solutions	- Testing and debugging	Generalization Evaluation	- -

As problem-solving is a cyclic activity, this study suggests for all the CT skills are divided into three algorithmic problem-solving stages, identified from the systematic literature review, to form a cyclic model for assessing CT skills (see Figure 7) in three algorithmic problem-solving stages.

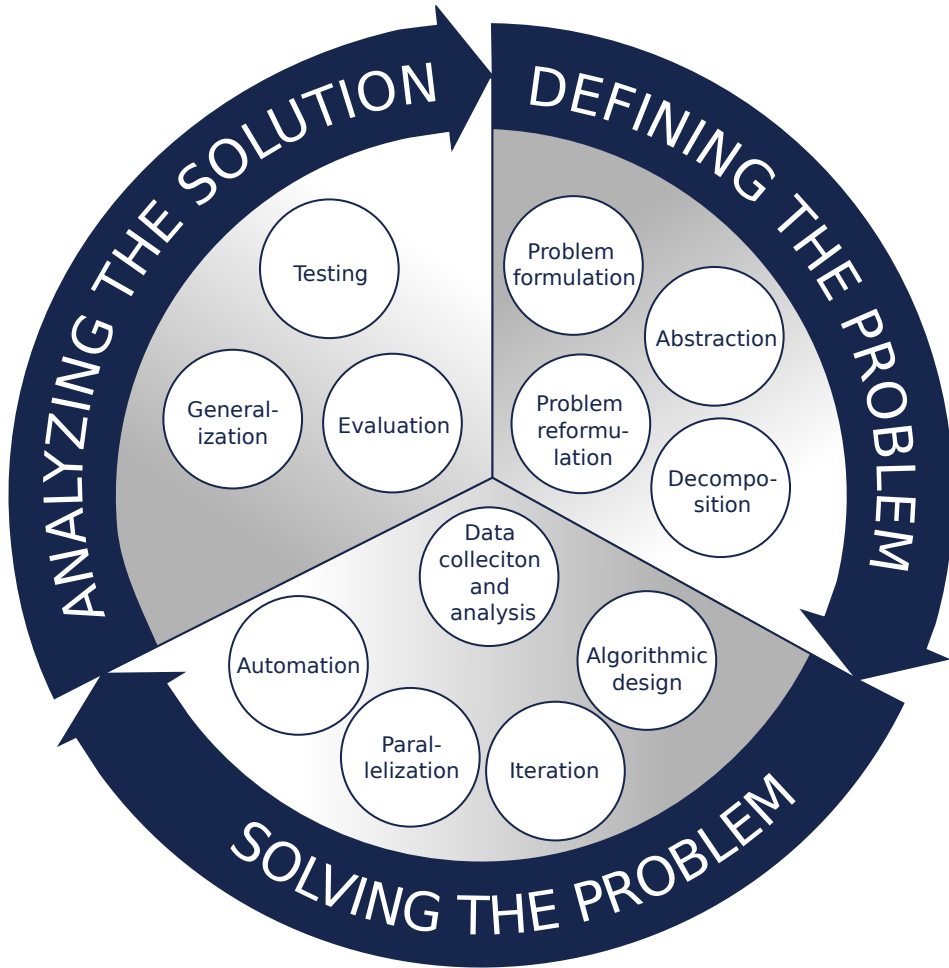


Figure 7. A new model for assessing CT skills

The next paragraphs describe these three stages together in the theoretical model for assessing CT skills created in this thesis.

4.2.1. Defining the Problem

The first stage of the new model for assessing CT is defining the problem, which derives from Wing's [Win06] description of CT as a way of solving problems. This stage of defining the problem includes all CT skills that are needed before actually starting to solve the algorithmic problem. Next, all the CT skills, included

in the stage of defining the problem, are described.

Problem Formulation. Although several authors do not include defining a problem as a separate skill, all articles describe it as part of the algorithmic problem-solving process. As CT is a thought process used in solving algorithmic problems, it is essential to understand and research the real-life problems that need to be solved. Problem formulation is a process of conceptualizing a problem verbally [RBE17]. This skill includes researching the problem. Decomposing and formulating the main problem is the first step that allows students to conceptualize, either verbally, e.g., by trying to formulate a question such as "How can I solve this problem?" or visually, e.g., by describing behaviors and relationships [PV17].

Abstraction. Secondly, all of the main articles presented in the table 1, include abstraction as an important part of CT. Abstraction occurs when a problem is formulated, and it is important to identify and extract relevant information to define the main idea(s) to solve the problem. As abstraction includes modeling the core aspects of complex problems or systems, it also includes modularizing the problem.

Solving an algorithmic problem starts with understanding the problem and defining how it can be solved by a computer, machine, or a human being. Abstraction has a vital part in it by being an ability to decide what details of a problem are important and what details can be ignored while solving the problem [SW13]. The ability to create and use abstractions is used constantly across mathematical and scientific undertakings, be it creating computational abstractions when writing a program, generating visualizations of data to communicate an idea or finding, defining the scope or scale of a problem, or creating models to explore further or understand a given phenomenon [Wei+16].

Although abstraction, to the idea of complexity, has been introduced as part of CT by Wing in her original article [Win08], the definition has developed over the subsequent years. Wing states that computing is the automation of our abstractions [Win08]. Abstraction is usually one of the first steps of problem-solving. Based on the descriptions [Win08], [SW13] and definitions [Bit], [Goo], [Cod], [Blo+15], tasks that need abstraction to include the following characteristics:

- tasks contain more information than is necessary for solving the task;
- the goal of the tasks is to decide what information is essential;
- the real-life problem needs to be understood and written down so that it can be solved.

In computer science, abstraction is used, for example, when a computer program plays chess or any other game filtering out the right moves and essential info for the right move. In cooking recipes, we name our ingredients and give the measurements giving the real-life objects names and values for further activities.

Problem Reformulation. Problem reformulation is reframing a problem into a solvable and familiar one [SSA17]. Problem reformulation is similar to problem formulation but is a skill connecting researching the real-life problem to visualiz-

ing the algorithms for a solution. This needs visual thinking and communicating of the problem to describe object interactions and metaphors that can support a suggested approach [PV17]. In a game creation project, problem reformulation can be a formulation of the story in terms of actors, virtual stage elements, game goals, rules, and characters' actions [Chi+17]. Problem reformulation needs a skill of abstraction [RBE17].

Decomposition. The fourth skill included in the first stage is the decomposition of the problem, which means breaking the problem down into manageable units. Decomposition is usually listed after the abstraction, and in this model, they follow each other. As problem formulation and reformulation have been added, this step of decomposition now follows problem formulation.

Most of the complex algorithmic problems need a skill of decomposition to break the problem down into smaller, more easily solved parts [SW13]. Already Polya (1948) provided a general discussion on the use of decomposition in problem-solving and Hertz (1964) [POL]. They claimed that with the decomposition principle, one could generally take account of more factors than when making direct or global judgments, and it would seem possible to analyze information on the various components more effectively by "computer" than by doing it in one's head.

Wing [Win08] has not used the word decomposition but has described breaking problems down by functionality as part of CT. Decomposition is required when dealing with extensive problems, complex systems, or complex tasks. Danny Edelson [Fow17] points out that creating solutions requires breaking problems down into chunks of functionality and sequencing them. Most recently, in refining his definition of CT, Guzdia [Guz08] also includes decomposition as a skill of CT. Based on the descriptions [Win08], [SW13], [Guz08], [Cou+11] and definitions [Bit], [Goo], [Cod], decomposition tasks include the following characteristics:

- tasks should be sufficiently tricky for not enabling solving in one piece;
- tasks should be solvable when broken down into smaller parts;
- the sequence of the steps of the solution should not be necessary because that already leads to the principle of algorithmic thinking skill.

In computer science, decomposition is used to break a computational graph problem into four sections, each completed by a different computer processor. Another example would be using several functions to conduct smaller tasks in a more complex code. In cooking recipes, we have several parts of the process: preparation, cooking, and decorating.

4.2.2. Solving the Problem

The second stage of the model for assessing CT skills includes aspects of solving the problem. This stage consists of all CT skills involved in creating the solution for the redefined algorithmic problem. Next, all the CT skills, included in the stage of solving the problem, are described.

Data Collection and Analyzis. The precondition for solving the problem algorithmically is collecting and analyzing data beforehand. Data collection and analysis include finding a data source for a problem area, collecting data from an experiment, write a program to do basic statistical calculations on a set of data, analyze data from an experiment, identify trends in data from statistics, use data structures such as an array, linked list, stack, queue, graph, hash table, etc., use histogram, pie chart, a bar chart to represent data, use sets, lists, graphs, etc., to contain data, summarize data from an experiment and represent trends [BS11]. Data representation is depicting and organizing data in appropriate graphs, charts, words, or images. [DBA17]

Algorithmic Thinking. After data manipulation, the solution for the problem is designed with algorithmic design (a series of ordered steps). Algorithmic thinking is evidenced through the creation of algorithms – algorithmic design. One of the most important skills of CT is algorithmic thinking. An algorithm is a method to solve a task (problem) that consists of defined instructions. Algorithmic thinking creates a step-by-step set of instructions carried out by a device [Voo+15], [SW13]. Algorithmic design is using the creation of algorithms and develops algorithmic thinking [SW13]. Algorithmic design relates to algorithmic thinking as the design of algorithms is part of many solution theories of operation research [AD16]. Algorithmic thinking and design is used not only in computer science but in other disciplines as well.

Based on the descriptions [SW13], [Cou+11] and definitions, [Bit], [Goo], [Cod], [Blo+15] algorithmic design tasks include the following characteristics:

- a problem that needs several steps to solve it;
- possible steps could be given beforehand;
- steps need to be used in the correct sequence.

In computer science, algorithmic thinking is the basis of solving problems in various disciplines: optimization and search algorithms to identify best chemicals for improving reaction conditions to improve yields in chemistry; quantum computing in physics; mathematics behind string theory; testing planes via computer simulations in engineering; electronic commerce in economics; crime scene investigations in law, etc. Most of these are using rather complex algorithms, but the logic behind them remains the same – each of the tasks needs a step-by-step solution that can be carried out by an information-processing agent. For example, in cooking recipes, the algorithmic design is the step-by-step instructions for creating the cake.

Iteration. Iteration is an adaptive process, one in which the plan might change in response to approaching a solution in small steps [RBE17]. Students are incremental and iterative while creating the paths with Logo as it gives them the opportunity to execute their commands and receive feedback immediately. [LK14]. Designing a project is not a clean, sequential process of first identifying a concept for a project, then developing a plan for the design, and then implementing the

design in code. Students use iterative cycles of imagining and building projects – developing a little bit, then trying it out, and then developing further, based on their experiences and new ideas [BR12].

Parallelization. Parallelization is threading, pipelining, dividing up data or tasks in such a way that it is processed parallel, e.g., running experiments with different parameters simultaneously [BS11]. Parallelism in a game creation is illustrated with the presence of two or more characters that can execute an action at the same time [FIC17]. In CT projects, events are represented by the listeners ("sensors") of actions that may be carried out. These events can be eventually evaluated by conditions. Conditional blocks can be added to evaluate a fact, after which a decision can be made. Data are present thanks to the possibility of integrating variables that can be managed by using logic and mathematic operators. Both data and operators provide inputs for conditional blocks as well [BR12]. In Scratch projects, parallelization occurs on two scripts when a message sending and receiving logic is applied [MRR15].

Automation. As algorithmic design uses parallelization and iteration, it eventually leads to the automation of the process. ISTE [IC11] views CT as algorithmic thinking with automation tools and data representation with the use of simulation. Automation is having computers or machines do repetitive tasks [DBA17]. It is the control flow realized with the help of control structures and information processing [Nie+17]. Students develop their own thinking way when they realize that the computers could produce more effective automatical solutions while solving the problems [KÇÖ17]. It can be an approach to problem-solving that can be automated in some way, and that can be applied in many subjects or areas [KÇÖ17]

4.2.3. Analyzing the Solution

The third stage of the model is finally analyzing the created solution. Next, all the CT skills included in the stage defining the problem are described.

Generalization. This stage includes generalization, which means transferring this problem-solving process to a broader range of problems. Generalization is the ability to express a problem solution in generic terms, which can be applied to different problems that share some of the same characteristics as the original [SW13]. Generalization is the ability to move from specific to broader applicability, such as understanding how to draw a square by defining internal angles and applying the same algorithm to produce an approximation of a circle. It is the ability to recognize parts of solutions that have been used in previous situations or that might be used in future cases. Generalization is the step of identifying how small pieces may be reused and reapplied to similar or unique problems. Based on the descriptions [SW13] and definitions [Bit], [Goo], [Cod], [Blo+15], generalization tasks include the following characteristics:

- the task to represent a problem in generic terms (e.g., function);

- using new or already familiar algorithms in new situations.

In computer science, generalization can be seen in the concept of "objects," which may contain data often known as attributes, and procedures, often known as methods. In cooking recipes, the structure of a recipe can be used for various recipes, and elements of one recipe can be used in other recipes.

Testing. An essential role in solving the algorithmic problem is evaluating and testing the solution, which includes analyzing the processes and the results in terms of efficiency and resource utilization. This stage also includes systematic testing and debugging, efficiency and performance constraints, error detection, etc. Debugging and performance testing are essential activities in CT learning [Lee+11]. As things rarely work just as imagined, it is critical for designers to develop strategies for dealing with – and anticipating – problems [BR12]. Various testing and debugging practices can be used, which are developed through trial and error, transfer from other activities, or support from knowledgeable others. Solutions can be tested by listing the various testing and debugging processes [BR12].

Evaluation. Evaluation analyzes processes to assess and recognize the processes and outcomes of efficiency and resource utilization [SW13]. Evaluation goes hand-in-hand with several of the elements of CT described above. In the context of solutions, analyzing could be interpreted as a similar term to evaluation. [GP18] Wing [Win08] expresses the need for a computational thinker to make trade-offs by evaluating the use of time and space, power, and storage. That leads to the idea that evaluation tasks often already need a solution for the problem to be analyzed in terms of whether the solution is correct and efficient. Based on the descriptions [Win08], [SW13] and definitions [Bit], [Goo], evaluation tasks include the following characteristics:

- solution for a problem is already given;
- task is to analyze if the solution is correct and efficient;

In computer science, if a device is needed to deliver something automatically to a customer, it needs to be programmable in an error-free, quick, safe, and straightforward way. In cooking, the easiest way of evaluating the cake is by looking at it and tasting it. Recipes can be changed in several ways: changing the ingredients, cooking time and temperature, or decorations. Efficiency can be altered by changing the procedure more efficiently by thinking about making more than one cake in a similar kitchen environment.

Problem-solving stages act cyclically as improvements can be made after all of the CT stages are completed. The solution can still be improved after the evaluation and testing by using again the formulation of the emerged problems. This way, the three-staged application of the CT model is repeated until the user is satisfied with the result in the current time, knowledge, and resources frame.

As these three problem-solving stages repeat in a cyclic manner, categorization of CT skills from six original articles allows us to create a new model for assessing

CT skills proposed by the author of the thesis (see Figure 7). This model covers all of the main dimensions extracted from the articles in a three-staged problem-solving cycle, relying on CT to solve problems algorithmically.

4.2.4. An Example of Using the New Model for Assessing CT Skills

To get a better overview of the new model for assessing CT skills, this section describes an example of the CT skills and stages, using an example of creating a plant watering system (see Figure 8).



Figure 8. A model for assessing CT skills with illustrations from the project measuring plants' soil humidity.

Plants tend to lose vitality due to excessive or insufficient watering. An example project to solve the problem of watering the plants at home starts with formulating the problem: How to create a plant watering system that reminds us when

to water the plant? The next step would be researching the problem. Abstraction occurs when filtering out the relevant information from the irrelevant by planning the way, how to collect data on soil moisture as a humidity percentage and to find out which would be the optimal soil humidity percentage for the specific plant. Extra data is necessary for turning on and off the alarm for saving the alarm clock state. Planned activities for solving the problem include measuring the soil humidity and turning the alarm on and off according to the data collected. Now, after the abstraction, the solution for the problem is decomposed into smaller solvable problems. Which hardware can be used for setting up the alarm and measuring the humidity? Which algorithms can be used to turn the alarm on and off according to the soil humidity percentage? An example of decomposition is dividing the process into smaller stages: connecting a humidity sensor as an input and an alarm clock as an output to the computer, creating algorithms for reading the data about soil humidity, and turning on and off the alarm accordingly. When the stage of defining the problem is over, the second stage starts (see Figure 8).

The second stage of the model includes the CT skills of solving the problem, which includes putting together the plant watering system. The Arduino platform can be used to control the soil humidity sensor and an alarm. The next step is to compose the algorithm for measuring the soil humidity percentage and turning on the alarm when humidity is too low. This algorithm of humidity regulation can be used for various plants by just adjusting the variables for the minimum level of humidity. An if-else loop can automatically turn on and off the alarm based on soil humidity.

The third stage is analyzing the solution. The CT skill of generalizing describes the applicability of the solution that can be expanded to other plants and situations like measuring room humidity, temperature, carbon dioxide, light, etc. This stage also includes evaluating the frequency of humidity readings, user notification, and the accuracy of data and performance of the system.

As solving a problem can be a cyclic process, after completing the third stage, the problem can be formulated again. For example, the user may not like an alarm clock for notification and may prefer LED lights or email notifications for that. If the user can not be near the plant to water it, or watering can be too labor-intensive, an automatic system can be built. Would it be possible to add an automated watering system safely with a water pump to the system? Can the design be more aesthetic by hiding the wires? Improving the solution can lead to real working solutions (see Figure 8).

Creating a new solution for the problem can lead to new problems. The second prototype can be improved by completing the three stages of CT skills again until the user is satisfied with the plant watering system.

4.3. Tasks for Assessing CT Skills at Basic School Level

The aim of study 2 of the thesis is to find and test an instrument to assess CT skills at basic school. To develop CT skills at basic school, a tool for assessing CT skills must be developed. Bebras challenge tasks have been suggested by Dagiene and Sentance [DS16] to assess and develop the skills of CT. Therefore, results from a Bebras Challenge were analyzed with CFA to identify four CT skills: abstraction, algorithmic design, decomposition, and evaluation. No tasks assessing the skill of generalization were included in the Bebras Challenge. There were 15 tasks in the Bebras challenge, whose factor loadings in the CFA came up as follows:

- Abstraction: .43, .36;
- Algorithmic thinking: .40, .34, .28, .29, .38, .36, .08, .20;
- Decomposition: .38, .43;
- Evaluation: .26, .23, .37.

Although the fit indexes of the model were almost satisfactory (TLI = .831, CFI = .865 and RMSEA = 0.037), all factor loadings were small (.08-.43), which means that the empirical study does not support the theory of the categorization of the tasks assessing the skills of CT accordingly. In addition, it appeared that two factors consisted of only two variables (tasks), while it is often expected that in a good model, each factor should consist of at least three variables.

As the CFA did not support the four-factorial model, in addition to CFA, EFA was used to determine if the Bebras challenge tasks could be divided into other dimensions that better describe the CT skills. As a result, a two-factorial model was suggested for assessing the skills of CT.

4.3.1. Exploratory Factor Analysis to Explore the Factor Structure of Bebras Challenge Tasks to Assess CT

As CFA gave relatively small factor loadings for the model, EFA with principal axis factoring was used with the Oblimin rotation and Kaiser normalization to determine if the Bebras challenge tasks could be divided into other dimensions that describe better the CT skills. As a result, a two-factorial model was suggested for assessing the skills of CT. Tasks describing the two factors are shown in Table 2, where Tabachnick and Fidell [TF07] recommend a minimum factor loading of .32 for including the task in the factor.

Table 2. Factor loadings of two factors of the tasks and predicted original skill assessed.

Task name	Factor 1	Factor 2	Original skill assessed
Geocaching	0.45	0.184	Abstraction
Crane operating	0.44	-0.13	Algorithmic thinking
Quick Beaver Code	0.41	0.20	Decomposition
Mushrooms	0.40	0.01	Algorithmic thinking
Biber Hotel	0.40	0.14	Algorithmic thinking
Robot the Stairs	0.39	0.20	Evaluation
Animation	0.33	0.25	Decomposition
Fair Share	0.33	0.31	Algorithmic thinking
Dream Dress	0.29	0.11	Algorithmic thinking
Bracelet	0.27	0.04	Evaluation
Cross Country	0.25	0.19	Algorithmic thinking
Animal Competition	0.20	0.44	Evaluation
Walnut Animals	0.31	0.40	Abstraction
Button Game	0.01	0.29	Algorithmic thinking
Pencil Alignment	0.17	0.20	Algorithmic thinking

Factor 1. Algorithmic thinking. The first factor, algorithmic thinking, includes mostly predictably algorithmic thinking tasks and some other tasks for assessing abstraction, decomposition, and even evaluation. Taking a closer look at the tasks, all of the tasks already include a description of the problem and the question. None of the tasks include skills of defining the problem and analyzing the solution. This way, all the algorithmic thinking tasks can be summarized by describing tasks that assess step-by-step solving skills. All the tasks used for assessing CT skills at the basic school level are in Appendix A.

Factor 2. Pattern recognition. The second factor, pattern recognition, includes tasks for predictably assessing evaluation, abstraction, and algorithmic thinking. Again, taking a closer look at the tasks, all of the tasks include a description of the problem and the question. None of the tasks include skills of defining the problem and analyzing the solution. This way, all the pattern recognition tasks can be summarized by including tasks that assess recognizing and applying familiar algorithmic patterns and logic. All the tasks used for assessing CT skills at the basic school level are in Appendix A.

As the Bebras challenge tasks are complex, and authors have admitted [DS16] that tasks may need various CT skills to solve, the results differ from the theory of assessing five skills of CT. The new model for assessing CT skills in study 1 (see Figure 7) is a theoretical model and in study 2 assessing four CT skills was empirically tested. A two-factorial model was suggested for the stage of solving the problem, which included two dimensions of CT skills:

- algorithmic thinking;
- pattern recognition.

Based on that, a new model for assessing CT is modified by combining the theoretical and empirical parts of the models by including algorithmic thinking and pattern recognition dimensions in the problem-solving stage by replacing data collection and analysis, parallelization, iteration, and automation (see Figure 9). The reasoning behind the replacement is that the skills of algorithmic thinking and pattern recognition are used in data collection and analysis, parallelization, iteration, and automation.

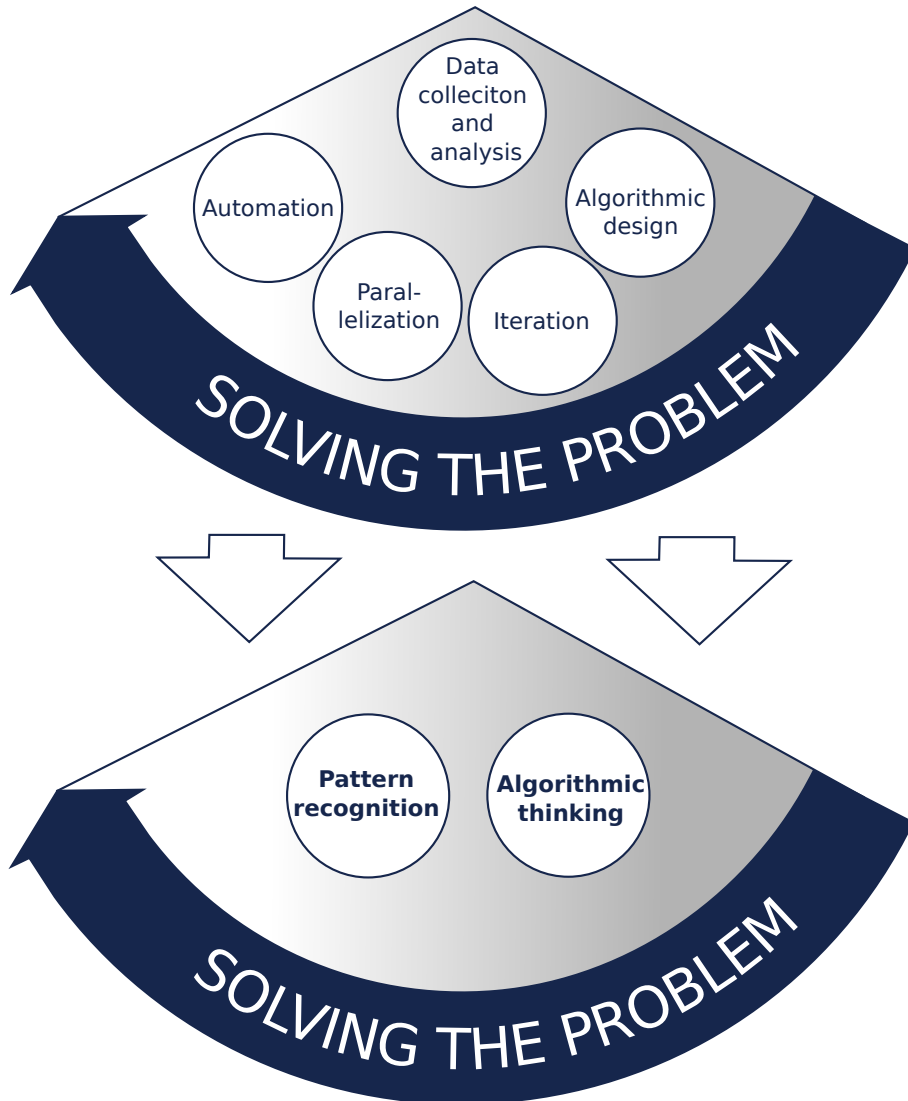


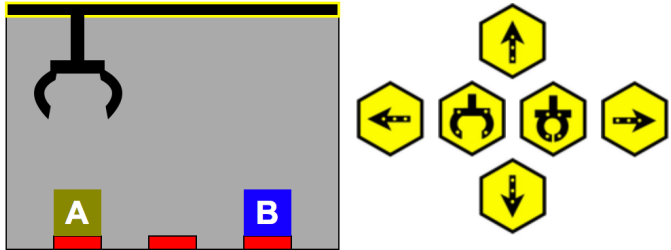
Figure 9. Solving the problem stage suggested for the model of assessing CT skills.

4.3.2. Description and an Example of Algorithmic Thinking Tasks

As previously mentioned, tasks included in Factor 1 can be characterized as tasks of algorithmic thinking. These tasks require step-by-step solutions for the problems. Examples of the tasks that assess algorithmic thinking are The Crane Operating (see Figure 10) and The Quick Beaver Code (see Figure 11).

The crane in the port of Lodgedam has six different input commands:

left
right
up
down
grab
let go



Crate A is in the left position, crate B is in the position on the right.

Question:

Using the command buttons, swap the position of the two crates.

The diagram shows a crane with a hook positioned above a port. Two crates, labeled A and B, are on the ground. Crate A is on the left and is green with a white 'A'. Crate B is on the right and is blue with a white 'B'. To the right of the crane are six yellow hexagonal buttons with black icons: an upward arrow, a leftward arrow, a rightward arrow, a downward arrow, and two buttons with crane hook icons (one with the hook open, one with the hook closed).

Figure 10. An example of algorithmic thinking task. Title: The Crane Operating

The first example of a task for assessing algorithmic thinking is The Crane Operating task (see Figure 10), where students had to switch the places of two crates using the given commands for the crane. This task needs a sequentially structured algorithm for exchanging two objects and was initially listed as an algorithmic design task. Places of two crates can only be changed if one of the crates is placed on extra empty space. The sensible way of doing this is starting from crate A. This task deals with designing and implementing sequential algorithms and process concepts. Most programs work with sequentially run operations, so each operation in the memory of the computer also needs extra space. As tasks in Berbas challenges are marked with three types of difficulty: A (easy), B (medium), and C (hard), this task is marked with B (medium) difficulty, and results show that 75% of students in the age group were able to solve this task correctly.

The second example of an algorithmic thinking task is The Quick Beaver Code (see Figure 11). Students had to use algorithmic thinking by looking at each square of the code and calculating the binary value of the square. However, this task was initially listed as a decomposition task. Although the answer can be filtered out without doing complicated calculations by looking only at the first square, which has the highest value of 365, on the top-left corner, it is still a step-by-step process to find the solution. QR codes are used in applications, including

The beavers want to encode numbers. They developed the Quick-Beaver-Code (QB-Code).

This is a code consisting of squares. Every square has a certain value. The squares are filled line by line from the bottom to the top and from right to left.

The value of the bottom right square is 1. The other squares have double the value of the square before them.

Example:

Here is a 3x3 QB-Code. The beavers have encoded a number by darkening some squares.

The number encoded is the sum of the values of the dark squares, so the number encoded in this QB-Code is $2 + 32 + 64 = 98$.

Question:
Of the following 4x4 QB-Codes, which one encodes the highest number?

A **B** **C** **D**

Figure 11. An example of an algorithmic thinking task. Title: The Quick Beaver Code

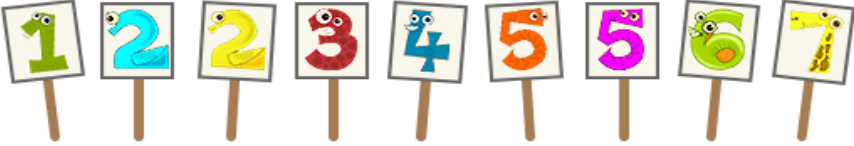
commercial tracking, entertainment, and transport ticketing, product/loyalty marketing, and in-store product labeling. Mobile phone users may receive text, add a contact to their device, open an URL, or compose a message after scanning a QR code. This task is marked with C (hard) difficulty, and results show that 48% of students solved this task correctly.

4.3.3. Description and an Example of Pattern Recognition Tasks

As previously mentioned, tasks included in Factor 2 can be characterized as tasks of pattern recognition. Students cannot solve these tasks by applying step-by-step solutions for solving and must recognize familiar patterns that emerge while solving the tasks. Examples of the tasks from Factor 2 are The Animal Competition task (see Figure 12) and The Button Game task (see Figure 13).

The first example of a task assessing pattern recognition is The Animal Competition (see Figure 12). This task requires an understanding of how data ordering rules are used. The conditions used in this task are called constraints. When working with data, some organization of the data is necessary. By solving this task, students can organize data logically, interpret patterns and models, break down problems into parts. This task is marked with B (medium) difficulty, and results show that 11% of students in the age group could solve this task correctly.

The second example of a pattern recognition task is The Button Game (see Figure 13), where students have to find out the minimal button moves to put all green

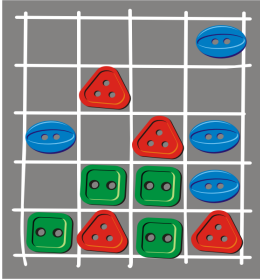


The beavers and dogs had a competition. In total nine animals took part. The nine participants had the following scores: 1, 2, 2, 3, 4, 5, 5, 6, 7. No dog scored more than any beaver. One dog tied with a beaver. There were also two other dogs that tied with each other.

Question:
How many dogs took part in the competition?
2, 3, 5, 6 or 7

Figure 12. An example of a pattern recognition task. Title: The Animal Competition

You can play this game on the ground. Draw a board and put the coloured buttons on the board. One step means to move one button to top, down, right or left through one box.



Question
What is the least number of steps to put all green squared buttons into one line at the bottom of the board?

Figure 13. An example of a pattern recognition task. Title: The Button Game

buttons into one line at the bottom of the board. This task cannot be solved easily by creating a step-by-step solution as the starting point is unclear, although it was initially listed as an algorithmic design task. Solving the task requires considering several solutions, which leads to the idea of finding the most optimal strategy from the beginning of the game. Having experience with that sort of algorithm is a benefit for finding the correct solution. This task includes solving algorithms but concentrates on optimization and logic. Optimization means finding the most cost-effective or highest achievable performance under given constraints by minimizing the undesired factors or maximizing desired ones. A strategy is needed

to find the best solution from all feasible solutions, as the most efficient way of solving this task is making a room first at the bottom of the board by moving blue buttons up. This task is marked with C (hard) difficulty, and results show that 6% of students in the age group could solve this task correctly.

These Bebras tasks are suggested for assessing CT. Still, they can also be used in formative assessment at schools so that teachers can track their students' progress in developing CT skills like algorithmic thinking and pattern recognition. Although this study shows that five CT skills cannot be distinguished, the tasks include applying CT skills like abstraction, decomposition, algorithmic thinking, evaluation, and generalization in various stages.

4.4. Tasks for Assessing CT Skills at Secondary School Level

To develop CT skills at secondary school, a tool for assessing CT skills must be developed. Results from study 2 show that Bebras challenge tasks could be used at the basic school level. In study 3, the basic school tasks were used to conduct a similar study at the secondary school level, replacing more manageable tasks with more complicated ones. CFA was used to confirm the theory of using the same instrument to assess two CT skills, algorithmic thinking and pattern recognition.

4.4.1. Pilot Study to Develop the Tasks to Assess CT Skills at Secondary School Level

Study 3 includes the pilot study and main study. Can these previously developed tasks for assessing CT skills at the basic school level be used at the secondary school level, too? This means filtering out more manageable tasks to be replaced with more difficult tasks more suitable for secondary school level. The tool for assessing CT skills in basic school was used in the pilot study of study 3, and results show that six tasks out of ten were too simple, having a successful completion rate of over 80%. As it is still important to remain focused on two skills of CT – algorithmic thinking and pattern recognition, these tasks were replaced with more difficult tasks from the same skills. Confirmatory factor analysis was used to confirm that all the tasks form a bi-factorial model according to the skills of CT.

4.4.2. Confirmatory Factor Model to Confirm the Factor Structure of Tasks to Assess CT Skills

After the pilot study in study 3, the main study of study 3 was conducted to confirm with the CFA the suggested dimensions of CT. The figure:13 shows the bi-factorial model as Factor 1 (f1) includes tasks that assess algorithmic thinking and Factor 2 (f2) to assess pattern recognition.

The graph in Figure 14 shows the factor loadings of the tasks 1–5 (t1-t5) in Factor 1 are from .429 to .745, and factor loadings of the tasks 6–10 (t6-t10) in

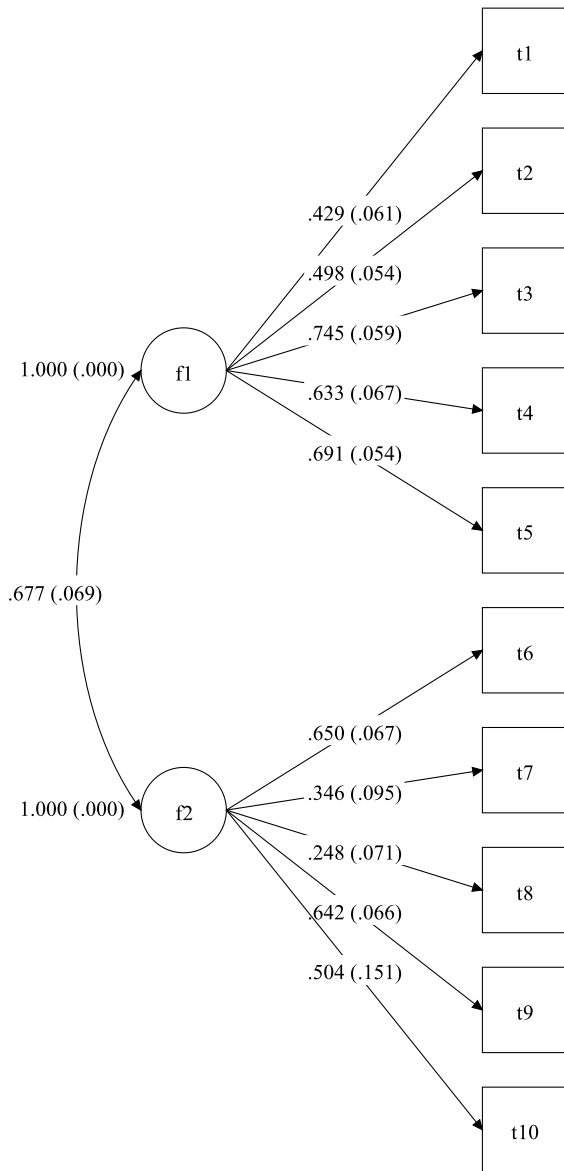


Figure 14. CFA model from the results of CT tasks (t1-t10) with the factor loadings of the two factors: algorithmic thinking (f1) and pattern recognition (f2).

Factor 2 are from .248 to .650. The correlation between the factors f1 (algorithmic thinking) and f2 (pattern recognition) is .677.

The fit indices of the model were good (TLI = 1.035, CFI = 1.000, RMSEA = .000, and WRMR = .541), which means that this empirical study supports the theory proposed that the tasks are divided into two dimensions being suitable for assessing the CT skills. CFI and TLI values are usually below one and are cur-

rently over the top. It can occur for smaller sample sizes [Mpl].

Although the individual factor loadings are not very high (.25-.75), Tabachnick and Fidell [TF07] recommend a minimum factor loading of .32 for including the task in the factor, and only task 8 had a value below that. Together, those tasks form a two-factorial model with a very high significance. The tasks can be suggested for formative assessment in lessons so that teachers can track their students' progress in developing CT skills.

All the tasks used for assessing CT skills at the secondary school level can be seen in Appendix B.

5. DISCUSSION

In a modernizing world, it is essential to develop CT skills as jobs are lost, and new jobs are created by automation [AGZ16]. CT skills are fundamental skills for everyone and belong to every child's analytical ability, just like reading, writing, and arithmetic [Win06]. Developing CT skills needs a systematic approach to assess the CT skills. The author of the thesis proposed a model for assessing CT skills and tools for assessing CT skills at basic and secondary school. To identify the CT skills to be assessed, research is needed to have a common understanding of CT skills, develop a model and an instrument to assess the CT skills.

This thesis offers a new model for developing CT skills based on six main studies identified from the systematic literature review:

1. Wing (2006) [Win06],
2. Barr and Stephenson (2011) [BS11],
3. CSTA and ISTE (2011) [IC11],
4. Brennan and Resnick (2012) [BR12],
5. Selby and Woollard (2013) [SW13], and
6. Moreno-León (2015) [MRR15].

Combining the theoretical new model for assessing CT skills from study 1 (see Figure 7) with the CT skills from the empirical study 2 (see Figure 9) and study 3 at basic and secondary school, a modified model for assessing CT skills is formed (see Figure 15).

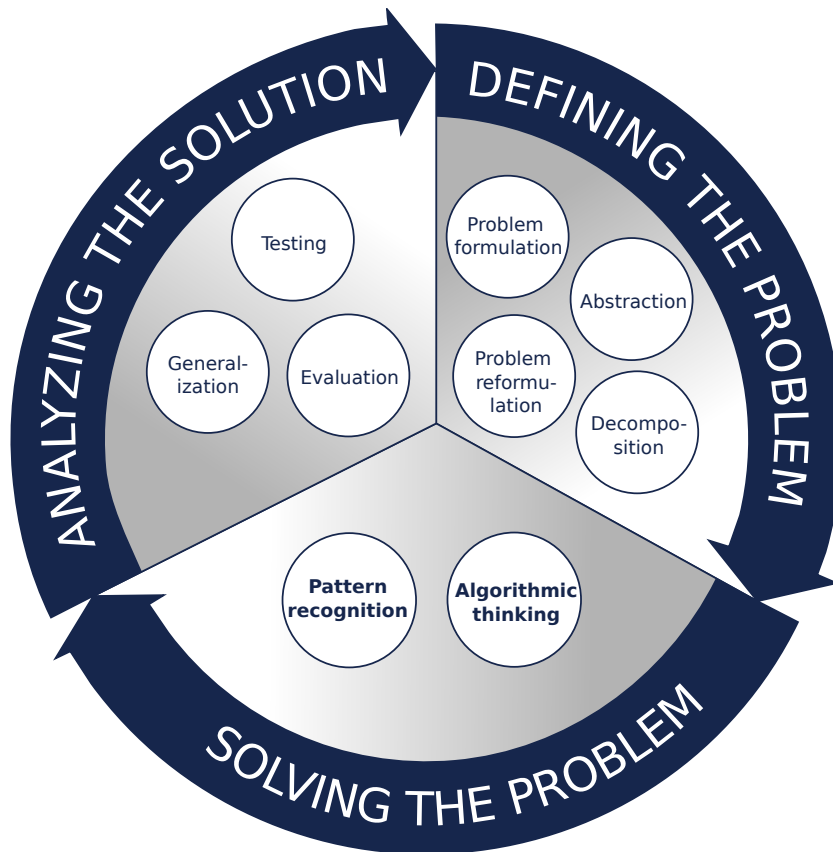


Figure 15. A modified model for assessing CT skills

This modified model is similar to the theoretical new model for assessing CT skills in three algorithmic problem-solving stages suggested by Reppening [RBE17], but the solving the problem stage is modified. The skills in the solving the problem stage are replaced with two new skills coming from analyzing Bebras Challenge tasks [Blo+15] suggested by Dagiene and Sentance [DS16] that were designed for assessing the CT skills. Empirical studies 2 and 3 in this thesis show that two main factors for categorizing the CT skills appeared from the factorial analysis: algorithmic thinking and pattern recognition. These two CT skills are used in data collection and analysis, parallelization, iteration, and automation.

Based on the systematic literature review, this thesis suggested the theoretical categorization of four CT skills in defining the problem stage. Categorization of the skills was not tested empirically, but are included in the new suggested model for assessing CT skills:

- problem formulation;
- abstraction;
- problem reformulation;
- decomposition.

The categorization of the skills in defining the problem stage is theoretical and should be empirically tested. Tasks and scenarios for developing CT skills should be created. It is a possibility that not that many skills can be identified in solving the problem stage. For example, to formulate a problem, the skill of abstraction is needed.

As this new model for assessing CT skills is a theoretical model, skills of the second problem-solving stage were empirically tested in study 2. Dagiene et al. (2016) [DS16] suggested Bebras Challenge tasks to assess abstraction, algorithmic design, decomposition, and evaluation. In this thesis, study 1 at the basic school level did not support that theory. Instead, a two-factorial model was suggested for the solving the problem stage, which included two dimensions of CT skills:

- algorithmic thinking;
- pattern recognition.

In this thesis, study 3 was conducted at the secondary school level to find out if tasks assessing CT skills at the basic school level could be used at the secondary school level. Assessing CT skills with a modified instrument for assessing CT skills at the secondary school level confirmed the theory of two dimensions suggested in the study conducted at the basic school level. The model for assessing CT skills was modified accordingly, combining the theoretical and empirical part of the problem-solving stage by replacing data collection and analysis, parallelization, iteration, and automation with algorithmic thinking and pattern recognition skills (see Figure 15).

This thesis did not test empirically the categorization of CT skills in the third analyzing the solution stage. The systematic literature identified four CT skills in the analyzing the solution stage:

- generalization;
- testing;
- evaluation.

Categorization of the skills in analyzing the solution stage is theoretical and should be empirically tested. Tasks and scenarios for developing CT skills should be created. It is possible that not that many skills can be identified in solving the problem stage. For example, to test the solution, the skill of evaluation is needed.

6. CONCLUSIONS AND IMPLICATIONS

6.1. Conclusions

Academic articles often include various definitions of CT. This Ph.D. thesis offers a unified model for a common understanding of the assessment of CT skills. RQ 1 and RQ 2. Which dimensions of CT skills can be identified in articles on developing CT? How can these dimensions from different articles be combined into a new theoretical model for developing CT?

- This thesis identified three stages of assessing CT skills as follows:
 - defining the problem, which includes the CT skills of formulating the problem, abstraction, problem reformulation, and decomposition;
 - solving the problem, which includes the CT skills of algorithmic thinking, and pattern recognition that lead to data collection and analysis, parallelization, iteration, and automation;
 - analyzing the solution, which includes the CT skills of generalization, testing, and evaluation.
- A new model for assessing CT skills (see Figure 15) is formed by combining the CT skills into one model. The model includes three sequential stages for solving algorithmic problems that could be used for assessing CT. Previous models of CT have not listed CT skills in the order of occurrence in such a cyclic manner, where each stage follows the previous one. This model can be used by teachers, students, and scientists to assess various project ideas and instructional activities.

RQ 3 and RQ 4. Which instrument can be used to assess CT skills at basic school? Which CT skills can be differentiated with an instrument used for assessing CT at basic school?

- This thesis suggests a test containing Bebras tasks for assessing CT skills at the basic school level. Two CT skills can be assessed in the stage of solving the problem in any kind of lesson, and the test is not limited to specific situations or software. Test results can be assessed either automatically online or manually using paper and pencil. Two factors that emerged from the study are characterized as algorithmic thinking and pattern recognition, which can be assessed by the test.

RQ 5 and RQ 6. Which instrument can be used to assess CT skills at secondary school? Which CT skills can be differentiated with an instrument used for assessing CT at secondary school?

- As an instrument for assessing CT skills at the basic school level was developed, the question remained, which instrument could be developed for assessing CT skills at the secondary school level. This thesis also presents a test that can assess CT skills at the secondary school level. The test consists

of 10 Bebras challenge tasks, where five tasks assess algorithmic thinking and five tasks pattern recognition skills. Results show that the two-factorial model is fitting. The main factors can be described as algorithmic thinking (tasks of finding step-by-step solutions for algorithmic problems) and pattern recognition (tasks of recognizing algorithms for solving algorithmic problems).

6.2. Implications

Practical implications:

- The main new aspects of the model for assessing CT skills include explaining the way of assessing the stages and skills of CT with examples from project-based learning. This model can be used in a problem-solving process to assess CT skills from start to finish by completing the sequential stages. This model can be used by teachers and students to assess and develop various project ideas and instructional activities.
- An instrument for assessing CT skills at basic school was created. This thesis suggests tasks for assessing the two CT skills (algorithmic thinking and pattern recognition) that can be used in any kind of basic school lessons; that is not limited to specific situations or software, and that can be effective in a paper or an online test that can be automatically assessed by the teacher or a researcher.
- An instrument for assessing CT skills at secondary school was created. This study suggests tasks for assessing two CT skills at secondary school; the tasks can be used in any kind of lesson; they are not limited to specific situations or software, and they can be used effectively in a paper on online tests. The instrument consists of 10 tasks with five tasks for each of the two CT skills: algorithmic thinking and pattern recognition.

Theoretical implications:

- A modified model for assessing CT skills was created, as previous models have not included CT skills in such a cyclic manner where the assessed CT skills are divided into three major problem-solving stages following each other.
- In assessing CT problem-solving skills at basic school, this study explored two main factors emerging, which can be characterized as algorithmic thinking (includes mainly tasks of algorithmic patterns and decomposition) and pattern recognition (includes mainly tasks of abstraction, generalization, and evaluation).
- In assessing CT problem-solving skills at secondary school, this study confirmed the two-factorial model is fitting. The two emerging main factors can be described as algorithmic thinking and pattern recognition, like in the basic school study.

Directions for further studies:

- Further research is needed to investigate the factorial structure of the modified model for assessing CT. More information about the factorial structure of the problem-solving stage could give us more information about the two-factorial structure of CT skills that could be assessed using other skills.
- Also, further research is needed to investigate in more depth each of the core CT skills and to create more tasks for assessing the various concepts of CT. Distinguishing other skills of CT to create tasks helps us understand their differences and use that knowledge in assessing and developing those skills. The relationships between the various skills could be further researched. The tasks are constructed so that they could be used in different subjects tackling computational problems. The scenarios and activities for integrating the tasks of CT into the basic and secondary school level classroom could be created, and the effectiveness of the activities developing the awareness and knowledge of the CT principles could be researched.

6.3. Limitations

The main limitations are as follows:

- The limitations of the study are that the tasks used in the test are relatively small and specific, which means that it can be difficult to create similar tasks. The tasks are only rated as true or false, but they could be developed further by allocating point scores to smaller parts of the tasks. Some tasks were still a little too easy for secondary school students and could be replaced with more complex tasks. Some tasks included multiple-choice questions, and, therefore, it was possible to select the right answer randomly. This study does not include scenarios for developing the CT skills that are assessed with the test, and, therefore, another systematic approach is required for that purpose.
- The limits of this model are that it is based only on the search term "computational thinking" and on the articles found in two search engines. Although these two search engines are rather significant, powerful, and well accepted in the research community, the new model does not include dimensions from the articles that do not present specific lists of CT skills. The identified clusters are based on a number of references from the other articles and can be changed in the future as new approaches get more references. Some aspects like abstraction, generalization, and data collection can occur in other sages, but the model only considered main occurrences, as explained.
- Further research is needed to create scenarios for developing and assessing the development of CT skills at various school levels. The tools for evaluating CT skills could be created to give us more information about the elements and the relations of the elements in the model. The model for

assessing CT skills can be used in various subjects and at various school levels for assessing the development of CT skills.

- Further research is needed to investigate the other dimensions of CT skills (defining the problem and analyzing the solution) to create tasks for developing and assessing those skills. The relationships between the various skills could be further researched. The tasks are constructed so that they can be used in different subjects tackling computational problems. Another idea for research would be developing activities for integrating the tasks of CT into the secondary school level classroom using various scenarios in lessons and verification of the effectiveness of such activities.

BIBLIOGRAPHY

- [AD16] Soumela Atmatzidou and Stavros Demetriadis. “Advancing students’ computational thinking skills through educational robotics: A study on age and gender relevant differences”. In: *Robotics and Autonomous Systems* 75 (2016), pp. 661–670.
- [AGZ16] Melanie Arntz, Terry Gregory, and Ulrich Zierahn. “The risk of automation for jobs in OECD countries: A comparative analysis”. In: (2016).
- [And16] Nicole D Anderson. “A call for computational thinking in undergraduate psychology”. In: *Psychology Learning & Teaching* 15.3 (2016), pp. 226–234.
- [Ang+16] Charoula Angeli et al. “A K-6 computational thinking curriculum framework: Implications for teacher knowledge”. In: *Journal of Educational Technology & Society* 19.3 (2016), pp. 47–57.
- [Ber+14] Marina Umaschi Bers et al. “Computational thinking and tinkering: Exploration of an early childhood robotics curriculum”. In: *Computers & Education* 72 (2014), pp. 145–157.
- [BG11] Natasha K Bowen and Shenyang Guo. *Structural equation modeling*. Oxford University Press, 2011.
- [Bit] BBC Bitesize. *KS3 Computer Science - Computational thinking*. URL: <http://www.bbc.co.uk/education/topics/z7tp34j>.
- [Blo+15] D Blokhuis et al. *UK Bebras Computational Thinking Challenge 2015. UK Bebras*. 2015. URL: <http://www.bebbras.uk/uploads/2/1/8/6/21861082/ukbebras2015-junioranswers.pdf>.
- [BMC17] Karen Selbach Borges, Crediné Silva de Menezes, and Léa da Cruz Fagundes. “The use of computational thinking in digital fabrication projects a case study from the cognitive perspective”. In: *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2017, pp. 1–6.
- [BR12] Karen Brennan and Mitchel Resnick. “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*. Vol. 1. 2012, p. 25.
- [Bra+17] Christian P Brackmann et al. “Development of computational thinking skills through unplugged activities in primary school”. In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. 2017, pp. 65–72.
- [BS11] Valerie Barr and Chris Stephenson. “Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?” In: *Acm Inroads* 2.1 (2011), pp. 48–54.

- [CCG17] Erick John Fidelis Costa, Livia Maria Rodrigues Sampaio Campos, and Dalton Dario Serey Guerrero. “Computational thinking in mathematics education: A joint approach to encourage problem-solving ability”. In: *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2017, pp. 1–8.
- [Che+17] Guanhua Chen et al. “Assessing elementary students’ computational thinking in everyday reasoning and robotics programming”. In: *Computers & Education* 109 (2017), pp. 162–175.
- [Chi+17] Giuseppe Chiazzese et al. “Promoting computational thinking and creativeness in primary school children”. In: *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*. 2017, pp. 1–7.
- [Cod] Code.org. *Computational Thinking*. URL: <https://code.org/curriculum/course3/1/Teacher>.
- [Cou+11] National Research Council et al. *Report of a workshop of pedagogical aspects of computational thinking committee for the workshops on computational thinking*. 2011.
- [Csi+15] Andrew Cszimadia et al. “Computational thinking—A guide for teachers”. In: (2015).
- [CTC17] Chih-Kai Chang, Yu-Tzu Tsai, and Ya-Lun Chin. “A visualization tool to support analyzing and evaluating Scratch projects”. In: *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE. 2017, pp. 498–502.
- [DBA17] Caitlin Duncan, Tim Bell, and James Atlas. “What do the teachers think? Introducing computational thinking in the primary school curriculum”. In: *Proceedings of the Nineteenth Australasian Computing Education Conference*. 2017, pp. 65–74.
- [Den17] Peter J Denning. “Remaining trouble spots with computational thinking”. In: *Communications of the ACM* 60.6 (2017), pp. 33–39.
- [DP16] Annwesa Dasgupta and Senay Purzer. “No Patterns in Pattern Recognition: A Systematic Literature Review”. In: *Eire, PA, USA: IEEE Press*, 2016, pp. 1–3. DOI: 10.1109/FIE.2016.7757676. URL: <https://doi.org/10.1109/FIE.2016.7757676>.
- [DS16] Valentina Dagiènè and Sue Sentance. “It’s computational thinking! Bebras tasks in the curriculum”. In: *International conference on informatics in schools: Situation, evolution, and perspectives*. Springer. 2016, pp. 28–39.
- [DS18] Hatice Yildiz Durak and Mustafa Saritepeci. “Analysis of the relation between computational thinking skills and various variables with the structural equation model”. In: *Computers & Education* 116 (2018), pp. 191–202.

- [DSS17] Valentina Dagienė, Sue Sentance, and Gabrielė Stupurienė. “Developing a two-dimensional categorization system for educational tasks in informatics”. In: *Informatica* 28.1 (2017), pp. 23–44.
- [DT19] Peter J Denning and Matti Tedre. *Computational thinking*. MIT Press, 2019.
- [Ein05] Albert Einstein. “Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]”. In: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: <http://dx.doi.org/10.1002/andp.19053221004>.
- [FIC17] Ilenia Fronza, Nabil El Ioini, and Luis Corral. “Teaching computational thinking using agile software engineering methods: A framework for middle schools”. In: *ACM Transactions on Computing Education (TOCE)* 17.4 (2017), pp. 1–28.
- [Fow17] Allan Fowler. “Engaging young learners in making games: an exploratory study”. In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 2017, pp. 1–5.
- [GBW13] Lindsey Gouws, Karen Bradshaw, and Peter Wentworth. “First year student performance in a test for computational thinking”. In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. 2013, pp. 271–277.
- [GMS93] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The L^AT_EX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.
- [Goo] Google. *Computational Thinking for Educators – Course*. URL: <https://computationalthinkingcourse.withgoogle.com>.
- [GP13] Shuchi Grover and Roy Pea. “Computational thinking in K–12: A review of the state of the field”. In: *Educational researcher* 42.1 (2013), pp. 38–43.
- [GP18] Shuchi Grover and Roy Pea. “Computational Thinking: A competency whose time has come”. In: *Computer science education: Perspectives on teaching and learning in school* 19 (2018).
- [Guz08] Mark Guzdial. “Education Paving the way for computational thinking”. In: *Communications of the ACM* 51.8 (2008), pp. 25–27.
- [Hof09] Leah Hoffmann. “Q&A The upper limit”. In: *Communications of the ACM* 52.1 (2009), 112–ff.
- [IC11] I ISTE and C CSTA. “Operational definition of computational thinking for K–12 education”. In: *National Science Foundation* (2011).
- [IST] ISTE. *Computational Thinking Competencies*. URL: <https://www.iste.org/standards/computational-thinking>.
- [KÇÖ17] Özgen Korkmaz, Recep Çakir, and M Yaşar Özden. “A validity and reliability study of the computational thinking scales (CTS)”. In: *Computers in human behavior* 72 (2017), pp. 558–569.

- [KGK16] Filiz Kalelioglu, Yasemin Gulbahar, and Volkan Kukul. “A framework for computational thinking based on a systematic research review”. In: (2016).
- [Knu] Donald Knuth. *Knuth: Computers and Typesetting*. URL: <http://www-cs-faculty.stanford.edu/%5C~%7B%7Duno/abcde.html>.
- [Kor+14] Külli Kori et al. “What influences students to study information and communication technology”. In: *INTED2014 Proceedings* (2014), pp. 1477–1486.
- [Kor+15] Külli Kori et al. “First-year Dropout in ICT Studies”. In: (2015), pp. 437–445.
- [Kot+17] Donna Kotsopoulos et al. “A pedagogical framework for computational thinking”. In: *Digital Experiences in Mathematics Education 3.2* (2017), pp. 154–171.
- [LB17] Tony Lowe and Sean Brophy. “An operationalized model for defining computational thinking”. In: *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2017, pp. 1–8.
- [Lee+11] Irene Lee et al. “Computational thinking for youth in practice”. In: *Acm Inroads 2.1* (2011), pp. 32–37.
- [Lep+16] Marina Lepp et al. “Self-and Automated Assessment in Programming MOOCs”. In: (2016), pp. 72–85.
- [Lep+17] Marina Lepp et al. “MOOC in Programming: A Success Story”. In: (2017), pp. 138–147.
- [Lep+18] Marina Lepp et al. “Troubleshooters for Tasks of Introductory Programming MOOCs”. In: *The International Review of Research in Open and Distributed Learning 19.4* (2018). DOI: <https://doi.org/10.19173/irrodl.v19i4.3639>.
- [Lib+09] Alessandro Liberati et al. “The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions explanation and elaboration”. In: *Journal of clinical epidemiology 62.10* (2009), e1–e34.
- [LK14] Sze Yee Lye and Joyce Hwee Ling Koh. “Review on teaching and learning of computational thinking through programming: What is next for K-12?” In: *Computers in Human Behavior 41* (2014), pp. 51–61.
- [Lui+18] Piret Luik et al. “Completion of Programming MOOC or Dropping Out: Are There Any Differences in Motivation”. In: (2018), pp. 329–337.
- [Lui+19a] Piret Luik et al. “Participants and Completers in Programming MOOCs”. In: *Education and Information Technologies 24.6* (2019), pp. 3689–3706. DOI: <https://doi.org/10.1007/s10639-019-09954-8>.

- [Lui+19b] Piret Luik et al. “What motivates enrolment in programming MOOCs?” In: *British Journal of Educational Technology* 50.1 (2019), pp. 153–165. DOI: <https://doi.org/10.1111/bjet.12600>.
- [Lui+20] Piret Luik et al. “Programming MOOCs – different learners and different motivation”. In: *International Journal of Lifelong Education* 39.3 (2020), pp. 305–318. DOI: <https://doi.org/10.1080/02601370.2020.1780329>.
- [Mar+18] Maria José Marcelino et al. “Learning computational thinking and scratch at distance”. In: *Computers in Human Behavior* 80 (2018), pp. 470–477.
- [MM98] Linda K Muthén and Bengt O Muthén. *Mplus: The comprehensive modeling program for applied researchers; user’s guide; [Version 1.0]*. Muthén & Muthén, 1998.
- [Mor+11] Ralph Morelli et al. “Can android app inventor bring computational thinking to k-12”. In: *Proc. 42nd ACM technical symposium on Computer science education (SIGCSE’11)*. 2011, pp. 1–6.
- [Mor+17] Jesús Moreno-León et al. “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*. 2017, pp. 2788–2795.
- [Mou+17] Chrystalla Mouza et al. “Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK)”. In: *Australasian Journal of Educational Technology* 33.3 (2017).
- [Mpl] Mplus. *Unusual TLI values*. URL: <https://www.statmodel.com/download/TLI.pdf>.
- [MRR15] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. “Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking”. In: *RED. Revista de Educación a Distancia* 46 (2015), pp. 1–23.
- [MRR16] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. “Comparing computational thinking development assessment scores with software complexity metrics”. In: *2016 IEEE global engineering education conference (EDUCON)*. IEEE. 2016, pp. 1040–1045.
- [MRR17] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. “Towards data-driven learning paths to develop computational thinking with scratch”. In: *IEEE Transactions on Emerging Topics in Computing* 8.1 (2017), pp. 193–205.
- [Mun+16] Roberto Munoz et al. “Game design workshop to develop computational thinking skills in teenagers with Autism Spectrum Disorders”.

- In: *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE. 2016, pp. 1–4.
- [Muu+20] Eerik Muuli et al. “Using image recognition to automatically assess programming tasks with graphical output”. In: *Education and Information Technologies* 25.6 (2020), pp. 5185–5203. DOI: <https://doi.org/10.1007/s10639-020-10218-z>.
- [Nie+17] Pia Niemelä et al. “Computational thinking as an emergent learning trajectory of mathematics”. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 2017, pp. 70–79.
- [Pap96] Seymour Papert. “An exploration in the space of mathematics educations.” In: *Int. J. Comput. Math. Learn.* 1.1 (1996), pp. 95–123.
- [Ped+17] Margus Pedaste et al. “What Happens to IT Education? The Case in Estonia with Some Recommendations for International Discussion”. In: *International Journal of Information and Education Technology* 7.3 (2017), p. 204.
- [Ped+19] Margus Pedaste et al. “Complex Problem Solving as a Construct of Inquiry, Computational Thinking and Mathematical Problem Solving”. In: 2161-377X (2019), pp. 227–231. DOI: <https://doi.org/10.1109/ICALT.2019.00071>.
- [POL] GEORG POLYA. “How to Solve It, Princeton, 1948”. In: *PolyaHow to Solve It1948* ().
- [PV17] Nikolaos Pellas and Spyridon Vosinakis. “How can a simulation game support the development of computational problem-solving strategies?” In: *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2017, pp. 1129–1136.
- [RBE17] Alexander Repenning, Ashok R Basawapatna, and Nora A Escherle. “Principles of computational thinking tools”. In: *Emerging research, practice, and policy on computational thinking*. Springer, 2017, pp. 291–305.
- [RFP14] Jonathan Francis Roscoe, Stephen Fearn, and Emma Posey. “Teaching computational thinking by playing games and building robots”. In: *2014 International Conference on Interactive Technologies and Games*. IEEE. 2014, pp. 9–12.
- [RHJ17] Simon Rose, Jacob Habgood, and Tim Jay. “An exploration of the role of visual programming tools in the development of young children’s computational thinking”. In: *Electronic journal of e-learning* 15.4 (2017), pp. 297–309.
- [Rob+17] Gregorio Robles et al. “Software clones in scratch projects: On the presence of copy-and-paste in computational thinking learning”. In: *2017 IEEE 11th International Workshop on Software Clones (IWSC)*. IEEE. 2017, pp. 1–7.

- [Rod+15] Jennifer A Rode et al. “From computational thinking to computational making”. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2015, pp. 239–250.
- [RPJ17] Marcos Román-González, Juan-Carlos Pérez-González, and Carmen Jiménez-Fernández. “Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test”. In: *Computers in human behavior* 72 (2017), pp. 678–691.
- [SDW14] Cynthia Selby, Mark Dorling, and John Woollard. “Evidence of assessing computational thinking”. In: (2014).
- [Sel15] Cynthia C Selby. “Relationships: computational thinking, pedagogy of programming, and Bloom’s Taxonomy”. In: *Proceedings of the workshop in primary and secondary computing education*. 2015, pp. 80–87.
- [SF13] Linda Seiter and Brendan Foreman. “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*. 2013, pp. 59–66.
- [Sol+16] Arash Soleimani et al. “A tangible, story-construction process employing spatial, computational-thinking”. In: *Proceedings of the The 15th International Conference on Interaction Design and Children*. 2016, pp. 157–166.
- [SSA17] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. “Demystifying computational thinking”. In: *Educational Research Review* 22 (2017), pp. 142–158.
- [SW13] Cynthia Selby and John Woollard. “Computational thinking: the developing definition”. In: (2013).
- [Tau+17] **Tauno Palts** et al. “Tasks for Assessing Skills of Computational Thinking”. In: *Informatics in Education*. 10th annual International Conference of Education, Research and Innovation (2017), pp. 2750–2759. DOI: <http://dx.doi.org/10.21125/iceri.2017.0784>.
- [TF07] Barbara G Tabachnick and Linda S Fidell. *Using multivariate statistics (5: e upplagan)*. 2007.
- [TP15] **Tauno Palts** and Margus Pedaste. “Model of Learning Computational Thinking”. In: (2015), pp. 211–221.
- [TP17] **Tauno Palts** and Margus Pedaste. “Tasks for Assessing Skills of Computational Thinking”. In: (2017), pp. 367–367.
- [TP19] **Tauno Palts** and Margus Pedaste. “Tasks for Assessing Computational Thinking Skills at Secondary School Level”. In: (2019), pp. 216–226. DOI: http://dx.doi.org/10.1007/978-3-030-35343-8_23.

- [TP20] **Tauno Palts** and Margus Pedaste. “A Model for Developing Computational Thinking Skills”. In: *Informatics in Education* 19.1 (2020), pp. 113–128. DOI: <http://dx.doi.org/10.15388/infedu.2020.06>.
- [Voo+15] Joke Voogt et al. “Computational thinking in compulsory education: Towards an agenda for research and practice”. In: *Education and Information Technologies* 20.4 (2015), pp. 715–728.
- [VT16] Michael Vallance and Phillip A Towndrow. “Pedagogic transformation, student-directed design and computational thinking”. In: *Pedagogies: An International Journal* 11.3 (2016), pp. 218–234.
- [Wei+16] David Weintrop et al. “Defining computational thinking for mathematics and science classrooms”. In: *Journal of Science Education and Technology* 25.1 (2016), pp. 127–147.
- [Win06] Jeannette M Wing. “Computational thinking”. In: *Communications of the ACM* 49.3 (2006), pp. 33–35.
- [Win08] Jeannette M Wing. “Computational thinking and thinking about computing”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1881 (2008), pp. 3717–3725.
- [Zho+16] Baichang Zhong et al. “An exploration of three-dimensional integrated assessment for computational thinking”. In: *Journal of Educational Computing Research* 53.4 (2016), pp. 562–590.

Appendix A. TASKS FOR ASSESSING CT AT BASIC SCHOOL LEVEL

A.1. Task 1. Geocaching

Two friends, Anna and Bob, are searching for treasure. They have a smartphone app that shows them the direction to the treasure they are looking for. The two boxes on the map show where the treasure is. Anna is searching for box 1. Bob is looking for box 2.



Anna and Bob are standing in the same place. The picture shows the map and a screenshot of the smartphones.

Question:

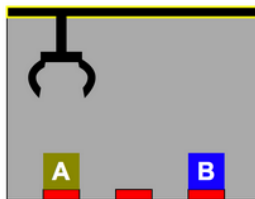
Where are Anna and Bob standing?



A.2. Task 2. Crane Operating

The crane in the port of Lodgedam has six different input commands:

left
right
up
down
grab
let go



Crate A is in the left position, crate B is in the position on the right.

Question:

Using the command buttons, swap the position of the two crates.

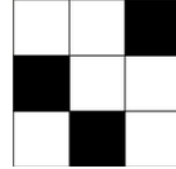
A.3. Task 3. Quick Beaver Code

The beavers want to encode numbers. They developed the Quick-Beaver-Code (QB-Code). This is a code consisting of squares. Every square has a certain value. The squares are filled line by line from the bottom to the top and from right to left. The value of the bottom right square is 1. The other squares have double the value of the square before them.

Example:

Here is a 3x3 QB-Code. The beavers have encoded a number by darkening some squares.

The number encoded is the sum of the values of the dark squares, so the number encoded in this QB-Code is $2 + 32 + 64 = 98$.



Question:

Of the following 4x4 QB-Codes, which one encodes the highest number?

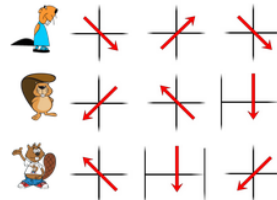
A	B	C	D

A.4. Task 4. Mushrooms

Three beavers are standing in a forest.

Each wants to go where there are mushrooms.

Arrows in the picture to the right show the directions the beavers will walk.

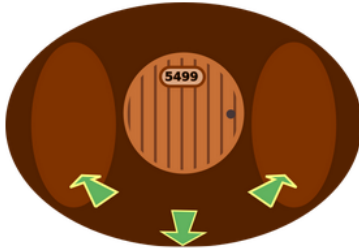


Question:

Where do the beavers end up?

A.5. Task 5. Biber Hotel

Over the years, the beavers constructed a huge beaver den with many, many rooms. The rooms are numbered and arranged in a particular tunnel structure. Click on the arrows in the picture to move through the den.

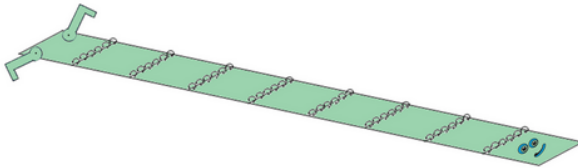


Question:

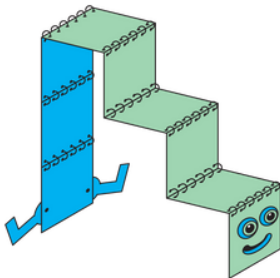
Find the room with number 1337. Click on 'Save' once you've found it.

A.6. Task 6. Robot the Stairs

Bob the Beaver is making a robot snake. The robot snake is constructed from identical square panels. Initially, Bob constructs the snake by laying out a row of square panels and connecting adjacent ones with hinges as shown in the picture below:



Bob can change the shape of the robot snake by bending it at its hinges. For example, Bob can transform the robot snake into some stairs. The stairs with 3 steps composed of a robot snake made from 9 square panels, is shown below:



Question:

How many square panels do we need to build stairs with height 7?

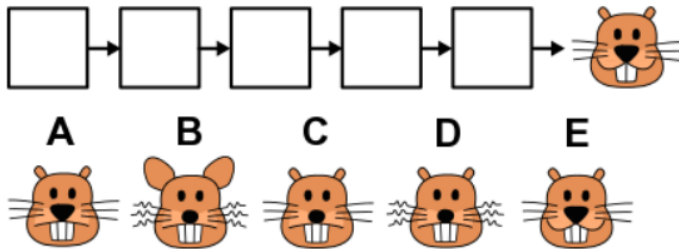
- A) 21
- B) 14
- C) 7
- D) 27

A.7. Task 7. Animation

Taro is planning an animation of a face that is made from a sequence of pictures.

To make the animation run smoothly, only one feature of the face should change from one picture to the next.

Unfortunately, the pictures got mixed up. Now Taro must find the correct order again. Luckily, he knows which picture is last.



Question:

Put the pictures in the correct order by dragging them onto the squares.

A.8. Task 8. Fair Share

Hamid has a 4 litre beaker full of a hazardous chemical.

Kazim has an empty 3 litre beaker and another empty 1 litre beaker.

Hamid and Kazim want to share the chemical between them equally and need a machine to do this safely.

The machine can pour one beaker in to another. It stops pouring when a beaker is completely emptied or filled, whichever happens first.

Question:

Find the sequence of pours that produces equal shares of the chemical for Hamid and Kazim.

Your sequence must use the minimum number of pours possible.

Start	Choose the <u>pours</u>	Build the sequence				
	<div style="border: 1px solid black; padding: 5px;"> <p>4 → 3</p> <p>4 → 1</p> <p>3 → 4</p> <p>3 → 1</p> <p>1 → 4</p> <p>1 → 3</p> </div>	<div style="border: 1px solid black; padding: 5px;"> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="height: 20px; width: 50px;"></td></tr> <tr><td style="height: 20px; width: 50px;"></td></tr> <tr><td style="height: 20px; width: 50px;"></td></tr> <tr><td style="height: 20px; width: 50px;"></td></tr> </table> </div>				

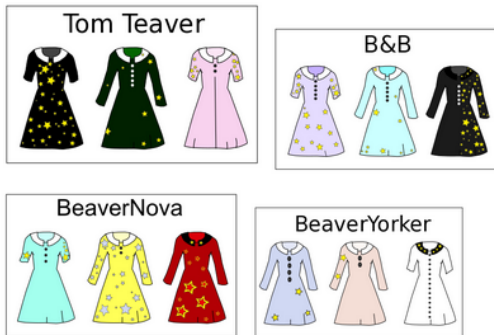
A.9. Task 9. Dream Dress

Kate wants to buy her dream dress.

It must have:

- short sleeves,
- more than 3 buttons,
- stars on its sleeves.

Four shops sell only the dresses shown.



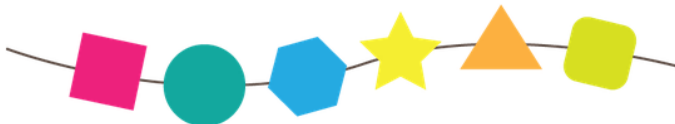
Question:

Which of these shops sells Kate's dream dress?

BeaverYorker, BeaverNova, B&B or TomTeaver

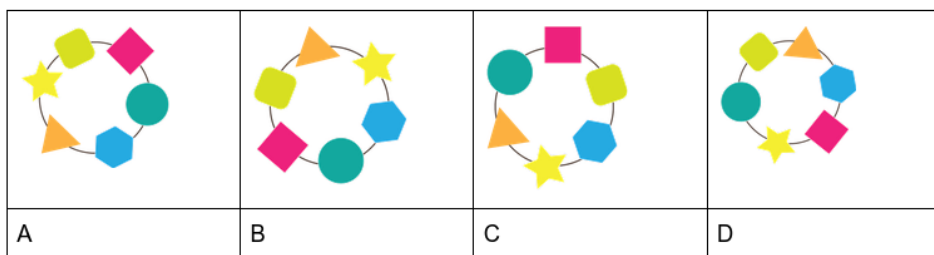
A.10. Task 10. Bracelet

Emily has broken her favourite bracelet. The broken bracelet now looks like this:



Question:

Which of the following four bracelets shows what the bracelet looked like when it was whole?



A.11. Task 11. Cross Country

Three very fast beavers will compete in a cross-country run.

Mr. Brown will overtake one beaver when running uphill.



Mrs. Pink will overtake one beaver when running downhill.

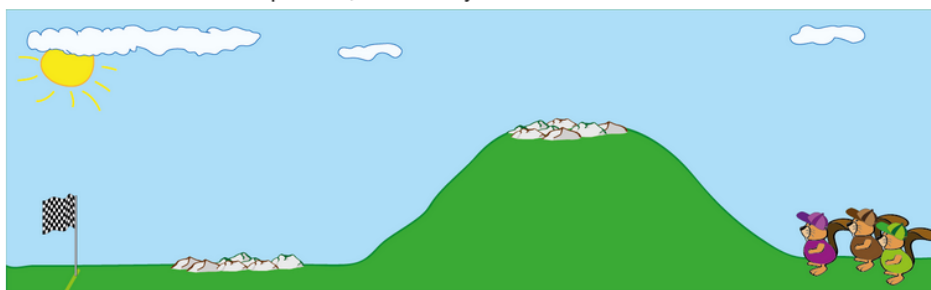


Mrs. Green will overtake one beaver when running across rocks.



The terrain is shown in the picture: uphill, followed by some rocks, downhill and then some more rocks.

Mrs. Pink starts in the first position, followed by Mr. Brown and Mrs. Green.



Question:

In which order will the beavers finish the race?

A.12. Task 12. Animal Competition



The beavers and dogs had a competition. In total nine animals took part.

The nine participants had the following scores: 1, 2, 2, 3, 4, 5, 5, 6, 7.

No dog scored more than any beaver.

One dog tied with a beaver.

There were also two other dogs that tied with each other.

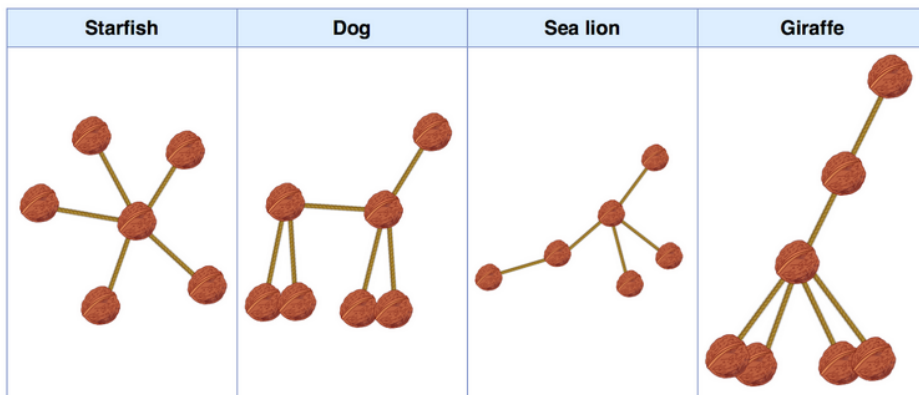
Question:

How many dogs took part in the competition?

2, 3, 5, 6 or 7

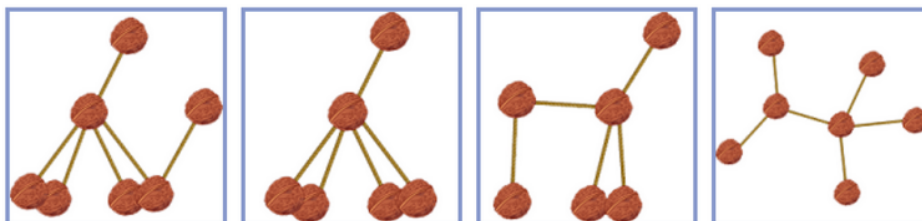
A.13. Task 13. Walnut Animals

Gerald was playing in the woods. He used nuts and sticks to create four nice animals. His sister managed to bend the animals around without removing any of the sticks. Gerald was very upset because he really loved the figure of a dog.



Question:

Which of the following figures can be bent back to make the figure of the dog again?

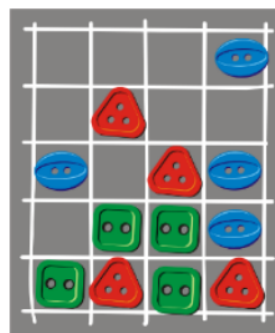


A.14. Task 14. Button Game

You can play this game on the ground. Draw a board and put the colored buttons. One step means to move one button to top, down, right or left through one box.

Question:

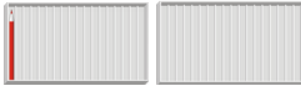
What is the least number of steps to put all green squared buttons into one line at the bottom of the board?



A.15. Task 15. Pencils Alignment

A little beaver is bored of drawing and wants to play with a box of pencils.

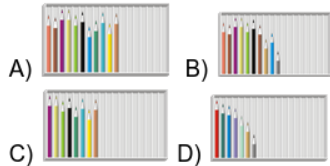
- Pencils are taken from the box one by one from left to right.
- Pencils are placed in Mom's and Dad's boxes, also from left to right.
- The first pencil is placed in Mom's box.
- Each other pencil is compared with the last pencil placed in Mom's box. If it is not longer than the last pencil placed in Mom's box, then it is also placed in Mom's box. Otherwise, it is placed in Dad's box.



Mom's box after first pencil is placed, Dad's box after first pencil is placed

Question

What will Dad's box look like after the little beaver places the last pencil?



Appendix B. TASKS FOR ASSESSING CT AT SECONDARY SCHOOL LEVEL

B.1. Task 1. Crane operating

The crane in the port of Lodgedam has six different input commands:

left
right
up
down
grab
let go



Box A is in the left position; box B is in the position on the right.

Question:

Using the command buttons, swap the position of the two boxes.

B.2. Task 2. Popularity

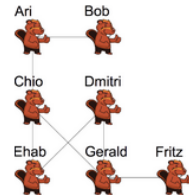
Seven beavers are in an online social network called Instadam. Instadam only allows them to see the photos on their own and their friends' pages.

In this diagram, if two beavers are friends they are joined by a line. After the summer holidays everybody posts a picture of themselves on all of their friends' pages.

Question:

Which beavers' picture will be seen the most?

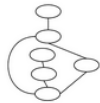
Ari, Bob, Chio, Dmitri, Ehab, Fritz or Gerald



B.3. Task 3. Word Chains

For his homework, Thomas had to write words on cards and connect them with rubber bands.

The teacher told him to connect any two words that differ by exactly one letter.



Thomas did this, as you can see in the picture on the right.

When Thomas returned from having a break he got a surprise.

Peter, his little brother, had erased all the words!

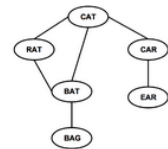
Also, the cards were completely mixed up, as you can see in the image on the left.

Importantly, the rubber bands still connected them as before.

Thomas was sure he could put the words back in the correct place.

Question:

Which of the pictures below contains the words in exactly the right places?



B.4. Task 4. Geocaching

Two friends, Anna and Bob, are searching for treasure. They have a smartphone app that shows them the direction to the treasure they are looking for. The two boxes on the map show where the treasure is. Anna is searching for box 1. Bob is looking for box 2. Anna and Bob are standing in the same place. The picture shows the map and a screenshot of the smartphones.



Question:



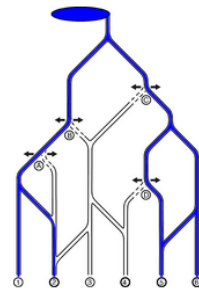
Where are Anna and Bob standing?

B.5. Task 5. Irrigation System

The beavers have created a clever irrigation system for their fields. The water flows from a lake at the top of the hill all the way down to the fields numbered 1 to 6 at the bottom.

Along the water canals, the beavers have installed four water gates A, B, C and D, where the water can only flow either to the left or to the right.

Answer and Explanation:



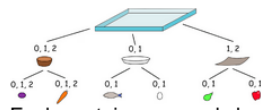
Question:

Which water gates should be changed, so that only the fields numbered 2, 4, 5 and 6 are irrigated?

B.6. Task 6. Beaver Lunch

Hm, what to take for lunch today?

The cafeteria gives instructions on how to choose a Beaver lunch.



This is shown as a diagram:

Below the tray you see different types of food containers.

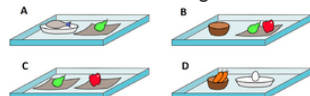
The numbers indicate how many containers of this type can be added to a tray.

Each container can only have food items put in it that are shown below it.

The numbers indicate how many food items of this type can be added to the containers.

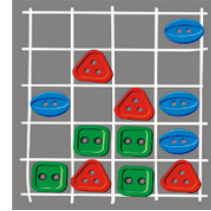
Question:

Which of the following lunches is not a proper Beaver lunch?



B.7. Task 7. Button Game

You can play this game on the ground. Draw a board and put the colored buttons. One step means to move one button to top, down, right or left through one box.



Question



What is the least number of steps to put all green squared buttons into one line at the bottom of the board?

B.8. Task 8. Decorating Chocolate

Everything is automated in a chocolate factory: the sweets are sliding on a conveyer, and there is a robot with a syringe, which draws different shapes.

The robot can perform these commands:



1. Leaf – draws: 
2. Circle – draws : 
3. Rotate k – rotates the sweet clockwise by k° .
4. Repeat n
[...]
] – repeats the commands inside brackets n times

For example, to perform

Repeat 4

[Circle

Rotate 90]



The robot will draw the flower as following:

Question

Which of the following command sequences the robot does NOT draw the flower?

<p>A. Repeat 6 [Rotate 30 Circle Rotate 30 Leaf]</p>	<p>B. Repeat 6 [Leaf Rotate 60] Rotate 330 Repeat 6 [Circle Rotate 300]</p>	<p>C. Repeat 6 [Leaf Rotate 60] Repeat 6 [Circle Rotate 60]</p>	<p>D. Repeat 3 [Rotate 120 Repeat 2 [Leaf Rotate 30 Circle Rotate 150]]</p>
---	---	---	--

B.9. Task 9. Pencils' Alignment

A little beaver is bored of drawing and wants to play with a box of pencils.

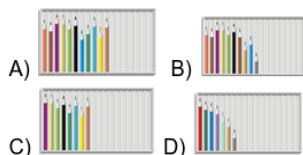
- Pencils are taken from the box one by one from left to right.
- Pencils are placed in Mom's and Dad's boxes, also from left to right.
- The first pencil is placed in Mom's box.
- Each other pencil is compared with the last pencil placed in Mom's box. If it is not longer than the last pencil placed in Mom's box, then it is also placed in Mom's box. Otherwise, it is placed in Dad's box.



Mom's box after first pencil is placed, Dad's box after first pencil is placed

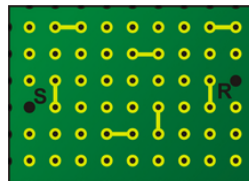
Question

What will Dad's box look like after the little beaver places the last pencil?



B.10. Task 10. Building a Chip

A small chip is composed of a grid of contacts (marked as dots). Some are already connected (marked as line segments). Connectors are always only between adjacent contacts, horizontally or vertically. We want to connect S and R with a continuous sequence of connectors, which do not touch any already connected contacts.



Question?

How many different ways are there to connect S and R with the least possible number of connectors?

SISUKOKKUVÕTE

Algoritmilise mõtlemise oskuste hindamise mudel

Tehnoloogia on kõikjal meie ümber ja arvutiteadus pole enam ainult eraldi distsipliin teadlastele, vaid omab aina laiemat rolli ka teistel aladel. Lugemise, kirjutamise ja arvutamise kõrval on algoritmilise mõtlemise (ingl *computational thinking*) oskus saamas vajalikuks oskuseks kõigile. Huvi algoritmilise mõtlemise arendamise vastu kasvab kõigil haridustasemetel alates eelkoolist kuni ülikoolini. Sellega seoses vajame aina enam üldhariduskoolide tasemel uuringuid, et omada paremat ülevaadet algoritmilise mõtlemise oskuste erinevatest dimensioonidest, et luua mudelit, mis aitaks praktiliselt hinnata algoritmilise mõtlemise oskuste taset.

Algoritmilist mõtlemist kirjeldatakse paljudes artiklites ja raportites, kuid sageli pole need artiklid omavahel kooskõlas ja puudub ühine arusaamine algoritmilise mõtlemise oskuste dimensioonidest, millele keskenduda arendamisel ja hindamisel. Doktoritöö sisaldab süstemaatilist kirjanduse analüüsi, kus mõjukamate artiklite sünteesimisel jõutakse kolmeetapilise algoritmilise mõtlemise oskuste mudelini. See mudel koosneb järgnevatest etappidest: i) probleemi defineerimine, ii) probleemi lahendamine ja iii) lahenduse analüüsimine. Need kolm etappi sisaldavad järgnevaid algoritmilise mõtlemise osaoskust: probleemi formuleerimine, abstrahheerimine, probleemi reformuleerimine, probleemi osadeks võtmine, andmete kogumine ja analüüs, algoritmiline disain, paralleliseerimine, itereerimine, automatiseerimine, üldistamine ning tulemuse hindamine ja testimine.

Selleks, et algoritmilist mõtlemist põhikoolis süstemaatiliselt arendada, on vaja mõõtevahendit vastavate oskuste mõõtmiseks põhikoolis. Üheks võimaluseks soovitatakse kasutada selleks rahvusvahelise informaatikaviktoriini Kobrase ülesandeid. Doktoritöö uurib testide abil empiirilisel, milliseid algoritmilise mõtlemise osaoskusi on võimalik eraldada Kobrase viktoriini tulemustest lähtuvalt. Uurimuslikku faktoranalüüsi kasutati 7100 osalejaga Kobrase viktoriini tulemuste analüüsimiseks ja selle põhjal loodi kahefaktoriline mudel, mille faktoreid saab nimetada üldnimetustega algoritmiline disain ja mustrite äratundmine.

Kuna loodud instrument töötab põhikoolis, siis tekkis küsimus, kas seda oleks võimalik kasutada ka gümnaasiumis? Doktoritöös esitletakse ka täiendatud ja empiirilisel testitud mõõtevahendit gümnaasiumi jaoks. 649 gümnaasiumiastme õpilast vastasid kohendatud testile ja kinnitava faktoranalüüsiga kinnitati, et täiendatud mõõtevahend on sobilik hindamiseks algoritmilise mõtlemise osaoskusi algoritmilist disaini ja mustrite äratundmist ka gümnaasiumis.

Doktoriöö arutelu peatükis pakutakse välja teoreetilisi ja empiirilisi tulemusi kokkuvõttev kohendatud algoritmilise mõtlemise oskusi hindav mudel.

PUBLICATIONS

CURRICULUM VITAE

Personal data

Name: Tauno Palts
Birth: 17.05.1985
Marital Status: Married
Contact: tauno.palts@ut.ee

Education

2014–2021 University of Tartu, PhD in Computer Science
2009–2011 University of Tartu, MA in Teacher of Mathematics and Informatics
2009–2011 University of Tartu, MA in Teacher of Informatics
2004–2009 University of Tartu, BSc in Information Technology
1992–2004 Kadrina Keskkool

Employment

2013– University of Tartu, assistant of informatics didactics
2018– Tartu Tamme Gymnasium, teacher of computer programming

Scientific work

Main fields of interest:

- developing computational thinking
- teaching programming
- teaching school robotics

ELULOOKIRJELDUS

Isikuandmed

Nimi: Tauno Palts
Sünniaeg: 17.05.1985
Perekonnaseis: abielus
Kontaktandmed: tauno.palts@ut.ee

Haridus

2014–2021 Tartu Ülikool, PhD informaatika
2009–2011 Tartu Ülikool, MA Matemaatika- ja informaatikaõpetaja
2009–2011 Tartu Ülikool, MA Informaatikaõpetaja
2004–2009 Tartu Ülikool, BSc Infotehnoloogia
1992–2004 Kadrina Keskkool

Teenistuskäik

2013– Tartu Ülikool, informaatika didaktika assistent
2018– Tartu Tamme Gümnaasium, programmeerimise õpetaja

Teadustegevus

Peamised uurimisvaldkonnad:

- algoritmilise mõtlemise arendamine probleemide lahendamisel
- programmeerimise õpetamine
- koolirobootika õpetamine

**DISSERTATIONES INFORMATICAЕ
PREVIOUSLY PUBLISHED IN
DISSERTATIONES MATHEMATICAE
UNIVERSITATIS TARTUENSIS**

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω -rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo.** Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Sor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAЕ UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh.** Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas.** Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi.** Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich.** Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka.** Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinmaa.** Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto.** Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.