

UNIVERSITY OF MACEDONIA
SCHOOL OF INFORMATION SCIENCES
DEPARTMENT OF APPLIED INFORMATICS

Computational Thinking through Programming: A conceptual
model for teaching and learning Computational Thinking

Ph.D. Dissertation

Christina Tikva

THESSALONIKI, GREECE
MAY, 2023

Computational Thinking through Programming: A conceptual model for teaching and learning Computational Thinking

Christina Tikva

Bachelor's Degree (BSc) in Applied Informatics, University of Macedonia, 2007
Master's Degree (MSc) in Computer Science and Management, Department of Computer Science of the Aristotle University, 2009

Ph.D. Dissertation

SUPERVISOR

Efthimios Tambouris

Professor, Department of Applied Informatics, University of Macedonia

ADVISORY COMMITTEE

Efthimios Tambouris

Professor, Department of Applied Informatics, University of Macedonia

Maria Satratzemi

Professor, Department of Applied Informatics, University of Macedonia

Nikolaos Fachantidis

Associate Professor, Department of Educational & Social Policy, University of Macedonia

EXAMINATION COMMITTEE

Efthimios Tambouris

Professor, Department of Applied Informatics, University of Macedonia

Maria Satratzemi

Professor, Department of Applied Informatics, University of Macedonia

Nikolaos Fachantidis

Associate Professor, Department of Educational & Social Policy, University of Macedonia

Stavros Demetriadis

Professor, Department of Informatics, Aristotle University of Thessaloniki

Spyridon Doukakis

Assistant Professor, Department of Informatics, Ionian University

Stylios Xinogalos

Professor, Department of Applied Informatics, University of Macedonia

Avgoustos Tsinakos

Professor, Department of Computer Science, International Hellenic University

Abstract

Computational Thinking (CT) through programming attracts increased attention as it is considered an ideal medium for the development of 21st century skills. CT initiatives have emerged around the world and there is a rapid increase in relevant research studies. The accumulation of research plethora leads to the need for a conceptual model of CT that could map the domain, facilitating comprehensive understanding of the domain's challenges. The aim of this thesis is a) to develop a conceptual model based on a systematic literature review that maps CT through programming in K-12 and higher education and b) to investigate the relationships between certain instances of the model, namely of the effects of scaffolding programming games and attitudes towards programming, on the development of students' Computational Thinking.

Regarding the first aim of this thesis, the proposed Computational Thinking through Programming in K-12 education (CTPK-12) conceptual model emerges from the synthesis of 101 studies and the identification of CT Areas. The proposed model consists of six CT Areas (namely Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building) and their relationships. Some of the relationships between areas have not yet been sufficiently explored in the literature such as which learning strategies enable the development of CT. The revised model for higher education is derived from a systematic mapping of 41 studies. This model includes the same CT Areas and relationships as CTPK-12, however it differs in sub-areas and instances. Knowledge Base Area, Assessment Area and Tools Area have significantly evolved throughout the years, while Capacity Building Area has only recently emerged. In addition, the introduction of CT to undergraduate students and preservice teachers differs mainly in the tools used and the CT elements that are assessed.

Regarding the second aim, students were introduced to CT under two distinct experimental conditions: a scaffolding version of a programming game and a non-scaffolding version of the same game. Results report statistically significant differences between the pre-intervention and post-intervention CT scores for all students and statistically significant improvement in learning outcomes in favor of the scaffolding group. In addition, the study hypothesized that attitudes towards programming would have

an impact on students' CT. Although this hypothesis has not been confirmed, the results suggest that students who have a less positive attitude towards programming could particularly benefit from scaffolding aspects in programming games.

Keywords: Computational Thinking, programming, K-12 education, higher education, scaffolding, computational thinking games, attitudes towards programming

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Professor Efthimios Tambouris for the continuous support of my Ph.D. studies, for his patience, motivation, and knowledge. His guidance helped me in all the time of research and writing of this thesis. I'd also like to extend my gratitude to Associate Professor Nikolaos Fachantidis and Professor Maya Satratzemi for their valuable and constructive advice.

Special mention goes to Dimitris Trakosas for his collaboration in publishing an article and to Maria Mousiou for the development of the Amazed scaffolding game (Mousiou 2021). I would also like to thank Marcos Román- González who shared with us the full version and the specification table sheet of the Computational Thinking Test (CTt) (Román-González et al., 2017).

Most importantly, none of this would have been possible without my family, my husband Ioannis and our two kids Euanthe and Vasilis. They have been a constant source of joy and fullness. Last but not least, I would like to thank my mother and father for supporting me always. I thank them for their unconditional love and the constant encouragement to continue and complete my studies.

Table of contents

1	Introduction	1
1.1	Problem definition and research objectives	1
1.2	Thesis Objectives and contribution	2
1.3	Thesis Contribution	5
1.4	Structure	6
1.5	Publications	7
2	Background Work	9
2.1	Introduction	9
2.2	Computational Thinking	9
2.3	Computational Thinking operational definitions	10
2.3.1	Angeli et al. (2016)	10
2.3.2	Barr & Stephenson (2011)	11
2.3.3	Brennan & Resnick (2012)	15
2.3.4	Csizmadia et al. (2015)	17
2.3.5	International Society for Technology in Education (ISTE) and Computer Science Teacher Association (CSTA) (2011)	22
2.3.6	Kalelioglu, Gulbahar, & Kukul (2016)	22
2.3.7	Selby (2013)	23
2.3.8	Shute et al. (2017)	24
2.3.9	Sondakh et al. (2020)	25

2.3.10	Weintrop et al. (2016)	26
2.3.11	Computational Thinking elements	27
2.4	Computational Thinking Literature Reviews	29
2.5	Summary	32
3	Methodology	33
3.1	Phase 1. Developing a Conceptual Model of Computational Thinking through programming in K-12 education (CTPK-12)	33
3.2	Phase 2. Extending the Computational Thinking through Programming in K-12 Education (CTPK-12) Conceptual Model for Higher Education	34
3.3	Phase 3. Designing and evaluating of a Scaffolding Computational Thinking game.	34
3.4	Phase 4. Investigating certain instances of the Learning Strategies and Factors CTPK-12 model's areas.	35
4	Computational Thinking through programming in K-12 Education (CTPK-12) Conceptual Model	36
4.1	Introduction	36
4.2	Study design	36
4.2.1	Study goal and research questions	36
4.2.2	Method	37
4.2.2.1	Elicitation of the domain knowledge	37
4.2.2.2	Visualization of the domain knowledge	41
4.2.3	Study limitations	42
4.3	The CTPK-12 model	42

4.3.1	CT Areas	46
4.3.1.1	Knowledge Base Area	46
4.3.1.2	Assessment Area	49
4.3.1.3	Learning Strategies Area	51
4.3.1.4	Factors Area	52
4.3.1.5	Tools Area	55
4.3.1.6	Capacity Building Area	56
4.3.2	CT Areas Relationships	58
4.4	Discussion	60
4.5	Summary	65
5	Extending the CTPK-12 model for higher education	66
5.1	Introduction	66
5.2	Study design	66
5.2.1	Study goal	66
5.2.2	Method	66
5.2.2.1	Definition of Research Questions	67
5.2.2.2	Conduct search for primary studies	67
5.2.2.3	Screening of studies	68
5.2.2.4	Classification Scheme Identification	68
5.2.2.5	Data extraction and mapping process	68
5.2.3	Study Limitations	69

5.3	Overview of Computational Thinking through programming studies in higher education.	69
5.3.1	Studies by year	69
5.3.2	Interventions for CT development in higher education.	70
5.4	The revised conceptual model for CT through programming in higher education (CTPHE)	74
5.4.1	CT areas in higher education	75
5.4.1.1	Knowledge Base	75
5.4.1.2	Learning Strategies	78
5.4.1.3	Tools	82
5.4.1.4	Assessment	85
5.4.1.5	Factors	88
5.4.1.6	Capacity Building	89
5.5	Discussion	90
5.6	Summary	94
6	Designing and evaluating a Computational Thinking tool	95
6.1	Introduction	95
6.2	Study design	95
6.2.1	Study goal and research questions	95
6.2.2	Research design and Participants	96
6.2.3	Instrument	96
6.2.4	Study Limitations	96

6.3	The aMazeD Scaffolding Computational Thinking Game	97
6.3.1	aMazeD General Description	97
6.3.2	Computational Thinking Concepts and Practices Covered by the Scaffolding Computational Thinking Game	99
6.3.3	aMazeD Scaffolding Features	100
6.3.4	aMazeD Analytics Features	102
6.4	Results	102
6.4.1	Demographic Data of the Participants	102
6.4.2	Perceived ease of use (PE)	103
6.4.3	Perceived usefulness (PU)	104
6.4.4	Attitude (AT)	106
6.4.5	Accessibility (AC)	107
6.4.6	Overall experience	108
6.5	Discussion	109
6.6	Summary	110
7	The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking	111
7.1	Introduction	111
7.2	Related Work	111
7.2.1	Scaffolding strategies in Computational Thinking research	113
7.2.2	Attitudes towards programming/Computer Science in Computational Thinking research	114

7.3	Study design	116
7.3.1	Study goal and research questions	116
7.3.2	Research design	117
7.3.3	Intervention instrument	118
7.3.4	Data collection	118
7.3.5	Study Limitations	118
7.4	Demographics	119
7.5	CTt	119
7.6	Analytics	119
7.7	Scale of Attitudes towards Programming	120
7.8	Does aMazeD have a positive impact on middle school students' CT development?	121
7.9	Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD version without scaffolding features?	121
7.10	Do attitudes towards programming have an impact on students' CT?	123
7.11	Do attitudes towards programming have an impact on students' CT improvement?	123
7.12	Discussion	124
7.13	Summary	129
8	Conclusions and direction for future research	130
8.1	Introduction	130
8.2	Conclusions Phase 1	130

8.3	Conclusions Phase 2	133
8.4	Conclusions Phase 3	133
8.5	Conclusions Phase 4	134
8.6	Limitations	134
8.7	Future work	135
	Appendixes	136
	Appendix A	136
	Appendix B	150
	Appendix C	157
	References	202

List of Figures

Figure 2-1. Computational Thinking dispositions (Barr & Stephenson, 2011)	14
Figure 2-2. Classroom Culture Characteristics (Barr & Stephenson, 2011).	14
Figure 2-3. Computational Thinking Concepts and Approaches (Csizmadia et al., 2015).	18
Figure 2-4. Definition of holistic Computational Thinking (Sondakh et al., 2020).....	26
Figure 2-5. Computational Thinking in Mathematics and Science definition (Weintrop et al., 2016).....	27
Figure 3-1. Method followed in this thesis.....	33
Figure 4-1. Method.....	37
Figure 4-2. Process applied for study selection adapted by Moher et al. (2009)	38
Figure 4-3. Example of elements recording and sub-areas identification.....	41
Figure 4-4. Example of evidence recording and relationship identification	41
Figure 4-5. Computational Thinking through Programming in K-12 education (CTPK-12) model.....	43
Figure 4-6. Percentage of studies by CT Areas to which they contribute in the periods 2006-2014 and 2015-2019. References to 2019 actually refer to period January 2019 to October 2019	45
Figure 4-7. Number of studies by CT element appearing more than twice in the examined studies.....	49
Figure 4-8. Number of studies by the most common learning strategies.....	52
Figure 4-9. Number of studies by tool	56
Figure 4-10. Example of a hypothesized research model based on CTPK-12 model.....	64

Figure 4-11. CTPK-12 model application in K-12 educational practice.....	65
Figure 5-1. Systematic mapping process, adapted from Petersen et al. (2008).....	67
Figure 5-2. Studies by year.....	70
Figure 5-3. Percentage of studies by branch in periods 2006-2016 and 2017-2020.....	73
Figure 5-4. The revised conceptual model for CT through programming in higher education (CTPHE)	74
Figure 5-5. Distribution of CT Knowledge Base elements sub-categories by time period	77
Figure 5-6. Distribution of learning strategies sub-categories by time period.....	81
Figure 5-7. Distribution of tools sub-categories by period.....	84
Figure 5-8. Distribution of assessment sub-categories by period	87
Figure 6-1. The aMazeD game environment.....	98
Figure 6-2. Semi-finished instructions	101
Figure 6-3. Demographic data of the participants	103
Figure 6-4. Results on PE1	103
Figure 6-5. Results on PE2.....	104
Figure 6-6. Results on PU1	104
Figure 6-7. Results on PU2	105
Figure 6-8. Results on PU3	105
Figure 6-9. Results on PU4	105
Figure 6-10. Results on PU5	106
Figure 6-11. Results on AT1	106

Figure 6-12. Results on AT2	107
Figure 6-13. Results on AC	107
Figure 7-1. Means of pre-tests scores by attitudes towards programming group	126
Figure 7-2. Means of score changes by attitudes towards programming group for the non-scaffolding group.....	127
Figure 7-3. Means of score changes by attitudes towards programming group for the scaffolding group.....	128

List of Tables

Table 2-1. Computational Thinking skills (Angeli et al.,2016)	10
Table 2-2. Computational thinking skills curriculum (Angeli et al., 2016)	10
Table 2-3. Computational Thinking operational definition (Barr & Stephenson, 2011) .	11
Table 2-4. Computational Thinking concepts, practices and perspectives (Brennan & Resnick, 2012).....	15
Table 2-5. Computational Thinking Concepts (Csizmadia et al., 2015).....	18
Table 2-6. Computational Thinking Techniques (Csizmadia et al.,2015)	21
Table 2-7. Framework for Computational Thinking as a Problem-Solving Process (Kalelioglu, Gulbahar, & Kukul, 2016).....	23
Table 2-8. Computational Thinking components (Shute et al., 2017)	24
Table 2-9. CT elements in CT operational definitions	28
Table 2-10. Literature Reviews in CT domain.....	30
Table 4-1. Approaches to Literature Reviews adopted from Webster and Watson (2002)	40
Table 4-2. CT Areas	43
Table 4-3. CT Areas' relationships	45
Table 4-4. Knowledge Base sub-areas	47
Table 4-5. Assessment sub-areas.....	50
Table 4-6. Learning strategies sub-areas	51
Table 4-7. Factors sub-areas.....	54
Table 4-8. Tools sub-areas	56

Table 4-9. Capacity Building sub-areas	58
Table 5-1. Inclusion and exclusion criteria	68
Table 5-2. Interventions for CT development in higher education	70
Table 5-3. Classification of branches	72
Table 5-4. CT Knowledge Base sub-categories	76
Table 5-5. Percentage of studies' CT Knowledge Base elements sub-categories by classified branch	77
Table 5-6. Learning strategies sub-categories	79
Table 5-7. Percentage of learning strategies sub-categories by classified branch	81
Table 5-8. Tools sub-categories	83
Table 5-9. Percentage of tools sub-categories by classified branch	84
Table 5-10. Assessment sub-categories.....	86
Table 5-11. Percentage of assessment sub-categories by classified branch.....	88
Table 5-12. Factors investigated in the selected studies.....	89
Table 5-13. Capacity Building methods.....	90
Table 6-1. CT Concepts and practices per aMazeD level	99
Table 7-1. Attitudes towards programming/CS found in the literature.....	115
Table 7-2. Distribution of participants by grade and gender.....	119
Table 7-3. Internal consistency of the scale of Attitudes towards Programming.....	121
Table 7-4. Computational Thinking pre-scores and post-scores means by game version	125

Table 7-5. Computational Thinking Conditional-Level, Loop-Level, Conditional-Use, Loop-Use scores, Conditional-Ratio and Loop-Ratio means by game version 125

Table 7-6. Computational Thinking changes in pre-scores and post-scores means by game version and attitudes towards programming group 128

1 Introduction

1.1 Problem definition and research objectives

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

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

Several literature reviews examine the whole domain from different perspectives as well as propose frameworks and definitions. Researchers in these studies review the literature in order to derive insights on CT through programming for K-12 curriculum (Lye & Koh, 2014), to understand the development and application of CT in education (Hsu et al. 2018), to facilitate CT learning and assessment within K-12 curricula (Shute, Sun, & Asbell-Clarke, 2017), to review CT literature in higher education (Czerkawski & Lyman, 2015) and to support educators in developing CT tasks and programs (Kalelioglu, Gulbahar, & Kukul, 2016). Despite all these efforts, a comprehensive mapping of the domain is still lacking. A comprehensive mapping of the domain would enable better understanding of challenges and guide future research.

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

When the research that has been conducted is mature and there is sufficient material, what often helps is the existence of a conceptual model. A conceptual model offers in developing domain understanding through aiding reasoning about the domain, communicating the domain details and documenting the domain for future reference (Gemino & Wand, 2004). In addition, a conceptual model could be an effective roadmap between what we know and what we need to know, providing a firm foundation for advancing the domain knowledge (Webster & Watson, 2002). Such a conceptual model of CT through programming education is still missing. Its existence could help researchers better understand the domain and its challenges through a holistic approach and identify areas that have already been covered by research and areas where more research is needed. In addition, a conceptual model serves as a point of agreement (Mylopoulos, 1992) and thus could support CT teaching and learning by providing a reference point for educators.

1.2 Thesis Objectives and contribution

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

Given that the bulk of the research concerns K-12 and that K-12 and higher education are often treated as related but distinct levels of education, we chose to base our model on K-12 and then extend it to higher education.

We also aim to further investigate certain instances of the model namely, Learning Strategies and Factors areas. To do so a) we design and evaluate a CT tool and b) we design and conduct an experimental study. In this context, the following objectives and research questions guide the work carried out in this dissertation:

O1: To develop a Conceptual Model of Computational Thinking through programming in K-12 education based on a Systematic Literature Review

This involves investigating and analysing the literature in order to elicit the areas of Computational Thinking domain and their relationships.

- **RQ1.1.** What are the areas of CT through programming in K-12 education domain?
- **RQ1.2** What are the sub-areas of each CT Area in K-12 education?
- **RQ 1.3** How do CT Areas relate to each other in K-12 education?

O2: To expand the Conceptual Model of Computational Thinking through programming in K-12 education for Higher Education

This involves studying the areas and relationships of the CTPK-12 conceptual model in the context of higher education. In addition, the analysis of these areas based on the following two dimensions is performed: a) their evolution over the years and b) the branches to which CT is applied.

- **RQ 2.1** What are the areas and sub-areas of teaching and learning CT through programming in higher education and how do they relate to each other?
- **RQ 2.2** How do these areas evolve over the years and how do they apply to various branches?

O3: To design, develop and evaluate a Scaffolding Computational Thinking game

This involves the design, implementation and evaluation of a Scaffolding CT game. In this dissertation, the design and evaluation of the game are presented, while the implementation was done by Maria Mousiou during her master thesis (Mousiou, 2021).

O3.1: To design the aMazeD Scaffolding Computational Thinking game

- **RQ 3.1.1** Which Computational Thinking concepts and perspectives should be covered by the aMazeD game?
- **RQ 3.1.2** Which scaffolding features should be included in the aMazeD game?
- **RQ 3.1.3** Which analytics should be included in the aMazeD game?

O3.2: To evaluate the aMazeD Scaffolding Computational Thinking game

- **RQ 3.2.1** Do students perceive the aMazeD Scaffolding Computational Thinking Game as ease to use?
- **RQ 3.2.2** Do students perceive the Scaffolding Computational Thinking Game aMazeD as effective on learning Computational Thinking?
- **RQ 3.2.3** Do students perceive the scaffolding features of the Scaffolding Computational Thinking Game aMazeD as effective in learning Computational Thinking?

O4: Using the CTPK-12 model to design and conduct an empirical study to investigate certain instances of the model namely, Learning Strategies and Factors areas.

This involves the investigation of the effects of a) scaffolding programming games and b) attitudes towards programming, on the development of middle school students' Computational Thinking.

- **RQ 4.1** Does the aMazeD programming game have a positive impact on middle school students' CT development?

- **RQ 4.2** Does the scaffolding version of the aMazeD programming game have a greater impact on middle school students' CT development than the version without scaffolding?
- **RQ 4.3** Do attitudes towards programming have an impact on middle school students' CT?
- **RQ 4.4** Do attitudes towards programming have an impact on middle school students' CT improvement?

1.3 Thesis Contribution

The development of a conceptual model for CT through programming aims to a) provide guidance to researchers in designing, delivering, and assessing CT studies and b) to provide guidance to educators in integrating CT into their educational practices. The benefits of utilizing such a conceptual model regard:

- Mapping the Computational Thinking through programming domain for future reference and communicating the domain details.
- Identifying areas that have already been covered by research and areas where more research is needed.
- Serving as a basis for future studies. In particular, the model could serve as a basis for hypothesized research models that establish a direct link between theory and statistical estimations.
- Providing evidence to teachers and policy-makers as well as bringing closer research, practice and policy.

Towards the aforementioned goals, the contribution is summarized in the following parts:

Developing a Computational Thinking through programming conceptual model: A conceptual model that presents the concepts and relationships of the domain and their visual representation. It comprises of six Computational Thinking Areas, namely *Knowledge Base Area*, *Assessment Area*, *Learning Strategies Area*, *Factors Area*, *Tools*

Area and *Capacity Building Area*. Each CT Area includes sub-areas that are populated with specific instances. Example of such sub-areas and instances include sub-areas of *Self-Report Methods*, *Tests*, *Artifact analysis*, *Observations* and *Assessment frameworks* in *Assessment Area* and instances of *scales*, *questionnaires*, *surveys*, *interviews*, *think-aloud protocol*, *journals* and *reflection reports* in *Self-Report* sub-area. Finally, we identify relationships between the CT Areas.

Utilizing the model to study certain CT Areas relationships: The model was utilized to explore a) the relationship between the instance of *Scaffolding (Learning Strategies Area)* and *CT concepts (Knowledge Base Area)* and b) the relationship between the instance of *Attitudes towards programming (Factors Area)* and *CT concepts (Knowledge Base Area)*. The effects of scaffolding programming games and students' attitudes towards programming on the development of students' Computational Thinking were explored. The implication of the study findings is important, as they provide support that scaffolding in CT games could be an effective strategy for teaching and learning CT to middle school students fostering a deeper understanding of CT concepts. In addition, when it comes to students' attitudes towards programming, students who perceive programming as less meaningful, less interesting and have lower programming self-efficacy could particularly benefit from scaffolding aspects in programming games.

1.4 Structure

The thesis is structured as follows:

The first chapter is the introduction of the research carried out, while the second chapter provides background information on related work. More specifically, it presents a study conducted on Computational Thinking definition frameworks in order to derive terms that describe the components of Computational Thinking that are repeatedly raised in the literature. In addition, the main characteristics of the literature reviews focusing on the field of Computational Thinking are briefly presented.

The third chapter presents the method followed during the Ph.D. research. Specifically, the research phases and the study designs that were followed are presented.

The fourth chapter presents the Computational Thinking through programming in K-12 (CTPK-12) education conceptual model that was designed based on a systematic literature review. In particular, the concepts and relationships of the model are presented, analysed and discussed in this chapter.

The fifth chapter presents the extension of the CTPK-12 model to include higher education. Similar to the previous chapter, the concepts and relationships of the model are presented and discussed, albeit in the context of higher education.

The sixth chapter presents the design and evaluation of a scaffolding programming game for Computational Thinking. In addition, the Computational Thinking concepts and practices covered by the game and the scaffolding framework on which it is based are presented.

The seventh chapter presents the investigation of certain relationships of the CTPK-12 model. Specifically, it presents an experimental study that exploits the scaffolding programming game presented in the previous chapter, to investigate the impact of the CPTK-12 model areas, namely “Learning Strategies” and “Factors”.

The eighth chapter offers conclusions drawn and future work.

1.5 Publications

The scientific findings of this thesis have been published in international journals as follows:

Chapter 4

Tikva, C., & Tambouris, E. (2021a). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, 162, 104083.

<https://doi.org/10.1016/j.compedu.2020.104083> [IF 11.82]

Chapter 5

Tikva, C., & Tambouris, E. (2021b). A systematic mapping study on teaching and learning Computational Thinking through programming in higher education. *Thinking Skills and Creativity*, *41*, 100849.

<https://doi.org/10.1016/j.tsc.2021.100849> [IF 3.652]

Chapter 7

Tikva, C., & Tambouris, E. (2022) The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking. *Education and Information Technologies*. [https://doi.org/10.1007/s10639-022-](https://doi.org/10.1007/s10639-022-11465-y)

[11465-y](https://doi.org/10.1007/s10639-022-11465-y) [IF 3.605]

Trakosas, D., Tikva, C., & Tambouris, E. Visual Programming and Computational Thinking Environments for K-9 Education: A Systematic Literature Review. *International Journal of Learning Technology* (accepted for publication).

2 Background Work

2.1 Introduction

The aim of this Chapter is to present definitions, related introductory concepts and previous work conducted in the field of Computational Thinking for a better understanding of the subject.

The remainder of this Chapter is organised as follows: Section 2.2 presents the definition of Computational Thinking according to previous research. Section 2.3 presents Computational Thinking frameworks that have been developed to provide an operational definition of Computational Thinking. The Section concludes with a summary of CT elements presented in the CT operational definitions. Section 2.4 presents literature reviews conducted in the field of Computational Thinking. Section 2.5 presents the summary of the Chapter.

2.2 Computational Thinking

Computational Thinking (CT) was firstly introduced by Papert (1980), who relates programming to procedural thinking skills. The term was reintroduced by Wing (2006) who defines CT as a process that “involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science” (Wing 2006, p.33). She points out that CT is a fundamental skill for everyone, not just for computer scientists and argues that “To writing and arithmetic, we should add CT to every child’s analytical ability” (Wing 2006, p.33). CT is the thought process that involves solving problems and designing model systems by utilizing Computer Science (CS) core concepts (Wing, 2008). Wing’s definition has subsequently become a reference point for discussion on CT. However, various other definitions have emerged in the literature (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013). Aho (2012) defines CT as “the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms.” Many other definitions exist in the literature.

CT definitions can be classified into two main categories: generic definitions that focus on CT as a thought process (Román-González, Pérez-González, & Jiménez-

Fernández, 2017) and operational definitions that describe what CT entails. The second category which comprises efforts that develop models describing CT elements, is presented in the next section.

2.3 Computational Thinking operational definitions

2.3.1 Angeli et al. (2016)

Angeli et al. (2016) propose a conceptual framework that describes CT skills (Table 2-1). Based on this framework they also present specific examples (Table 2-2) for each CT skill taking into account the age of the students.

Table 2-1. Computational Thinking skills (Angeli et al.,2016)

Skill	Definition
Abstraction	The skill to decide what information about an entity/object to keep and what to ignore (Wing, 2011).
Generalization	The skill to formulate a solution in generic terms so that it can be applied to different problems (Selby, 2014).
Decomposition	The skill to break a complex problem into smaller parts that are easier to understand and solve (National Research Council, 2010; Wing, 2011).
Algorithms Sequencing Flow of control	The skill to devise a step-by-step set of operations/actions of how to go about solving a problem (Selby, 2014). The skill to put actions in the correct sequence (Selby, 2014). The order in which instructions/actions are executed (Selby, 2014).
Debugging	The skill to identify, remove, and fix errors (Selby, 2014).

Table 2-2. Computational thinking skills curriculum (Angeli et al., 2016)

Skill	K-2 (ages 6 to 8)	3-4 (ages 9 to 10)	5-6 (ages 11 to 12)
Abstraction	With the use of external reference systems, create a model/representation to solve a problem.	Create a model/representation to solve a problem	Create a new model/representation to solve a problem.
Generalization	Identify common patterns between older and newer problem-solving tasks, and use sequences of instructions previously employed, to solve a new problem	Remix and reuse (by extending if needed) resources that were previously created	Remix and reuse (by extending if needed) resources that were previously created.

Decomposition	Break a complex task into a series of simpler subtasks	Break a complex task into simpler subtasks. Develop a solution by assembling together collections of smaller parts.	Break a complex task into simpler subtasks. Develop a solution by assembling together collections of smaller parts.
Algorithmic thinking	Define a series of steps for a solution. Put instructions in the correct sequence.	Define a series of steps for a solution. Put instructions in the correct sequence. Repeat the sequence several times (iteration).	Define a series of steps for a solution. Put instructions in the correct sequence. Repeat the sequence several times (iteration). Make decisions based on conditions. Store, retrieve, and update variables. Formulate mathematical and logical expressions
Debugging	Recognize when instructions do not correspond to actions. Remove and fix errors.	Recognize when instructions do not correspond to actions. Remove and fix errors.	Recognize when instructions do not correspond to actions. Remove and fix errors.

2.3.2 Barr & Stephenson (2011)

Barr & Stephenson's (2011) presents CT concepts and capabilities in the context of K-12 education. The framework practically approaches the integration of Computational Thinking in K-12 classrooms by providing tangible examples (Table 2-3).

Table 2-3. Computational Thinking operational definition (Barr & Stephenson, 2011)

CT Concept, Capability	CS	Math	Science	Social Studies	Language Arts
------------------------	----	------	---------	----------------	---------------

data collection	find a data source for a problem area	source for a problem area doing probability exercises, for example, flipping coins or throwing dice	collect data from an experiment	study battle statistics, or population data	do linguistic analysis of sentences
data analysis	write a program to do basic statistical calculations on a set of data	count occurrences of flips, dice throws and analyzing results	analyze data from an experiment	identify trends in the data from the statistics	identify patterns for different sentence types
data representation and analysis	use data structures such as array, linked list, stack, queue, graph, hash table, etc.	use a histogram, pie chart, bar chart, etc. to represent data; use sets, lists, graphs, etc. to contain data	summarize data from an experiment	summarize and represent the trends	represent patterns of different sentence types
abstraction	use procedures to encapsulate a set of often repeated commands that perform a function	use variables in Algebra; identifying essential facts in a word problem	build a model of a physical entity	summarize facts; deduced conclusions from facts	use of simile and metaphor
analysis and model validation	validate random number generator	curve fitting	validate that the model is correct		
automation		use tools such as: Geometer Sketch Pad; Star Logo; Python code snippets	use Prove ware	use Excel	use a spell checker
testing and verification	debug a program; wire unit tests; formal program verification	do guess and check	validate and clean data		

algorithms & procedures	study classic algorithms; implement an algorithm for a problem area	do long division, factoring; do carries in addition/subtraction	do an experimental procedure		write instructions
problem decomposition	define objects and methods; define main and functions	apply order of operations in an expression	do a species classification		write an outline
control structures	use conditionals, loops, recursion, etc.	study functions in algebra compared to functions in programming; use iteration to solve word problems			write a story with branches
parallelization	threading, pipelining, dividing up data or task in such a way to be processed in parallel	solve linear systems; do matrix multiplication	run experiments simultaneously with different parameters		
simulation	algorithm animation, parameter sweeping	graph a function in a Cartesian plane and modify values of the variables	simulate movement of the solar system	play Age of Empires; Oregon	Trail do a re-enactment from a story

The framework also includes dispositions (Figure 2-1) and characteristics of a classroom culture (Figure 2-2) that could contribute to the development of Computational Thinking.

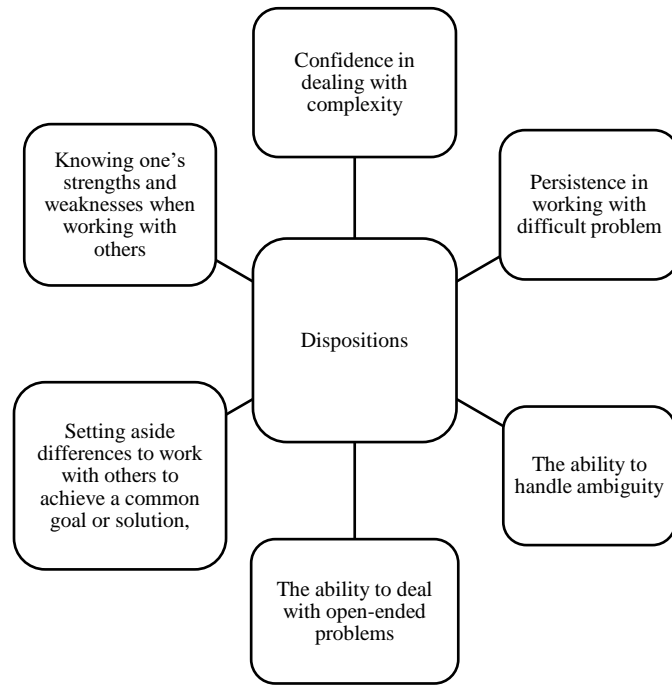


Figure 2-1. Computational Thinking dispositions (Barr & Stephenson, 2011)

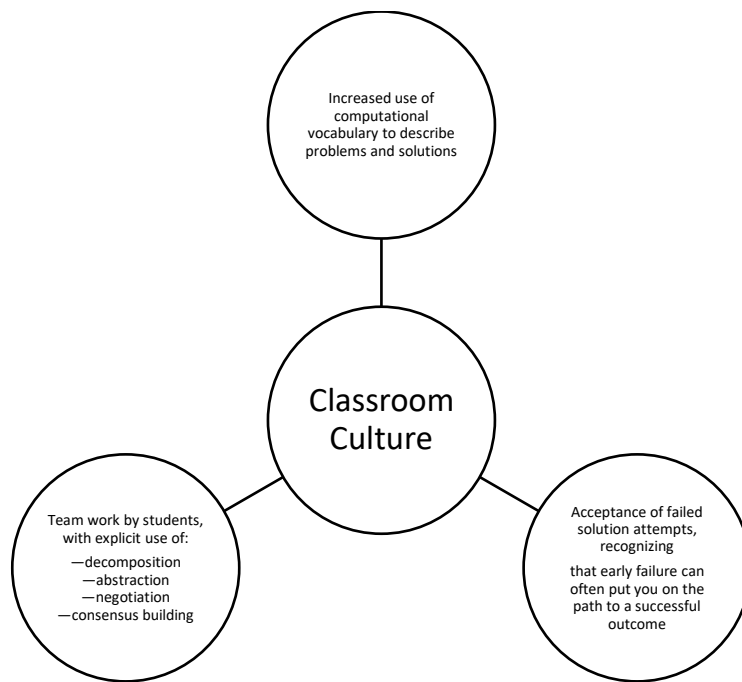


Figure 2-2. Classroom Culture Characteristics (Barr & Stephenson, 2011).

2.3.3 Brennan & Resnick (2012)

Brennan and Resnick (2012) introduced a Computational Thinking framework that describes CT concepts, practices and perspectives for young learners using Scratch to design applications.

According to this framework there are three dimensions of Computational Thinking (Table 2-4):

- Computational Concepts, the concepts that students use as they program.
- Computational Practices, the practices that students develop as they program.
- Computational Perceptions, the perceptions that students form about the world around them and about themselves.

Table 2-4. Computational Thinking concepts, practices and perspectives (Brennan & Resnick, 2012)

Dimension	Element	Description
Computational Thinking Concepts	Sequences	A series of individual steps or instructions that can be executed by the computer.
	Loops	A mechanism for running the same sequence multiple times.
	Parallelism	Sequences of instructions happening at the same time.
	Events	One thing causing another thing to happen – essential component of interactive media.
	Conditionals	The ability to make decisions based on certain conditions, which supports the expression of multiple outcomes.
	Operators	Operators provide support for mathematical, logical, and string

		expressions, enabling the programmer to perform numeric and string manipulations.
	Data	Data involves storing, retrieving, and updating values.
Computational Practices	Being incremental and iterative	The design and implementation of a project is an evolutionary process. It consists of iterative cycles of design, development and execution of the program and its further development, based on the experiences gained and new ideas.
	Testing and debugging	Development of strategies for dealing with and anticipating problems in the development of a project.
	Reusing and remixing	Reuse, modify and mix projects created by others to create a more complex project that would not otherwise be possible.
	Abstraction and modularizing	Creating a large project by combining collections from smaller sections. For example, code segmentation depending on the functionality of the commands.
Computational Perspectives	Expressing	The use of technology not only as a consumer but as a means of design and expression.
	Connecting	Interact with others in the context of social learning practice.
	Questioning	Being encouraged to ask questions and challenge the obvious, in some cases, answering these questions with suggestions and designs.

2.3.4 Csizmadia et al. (2015)

Csizmadia et al. (2015) suggest “Computational thinking is a cognitive or thought process involving logical reasoning by which problems are solved and artefacts, procedures and systems are better understood”. In addition, they propose a conceptual framework that describes Computational Thinking Concepts, Approaches (Figure 2-3) (Table 2-5) and Techniques (Table 2-6).

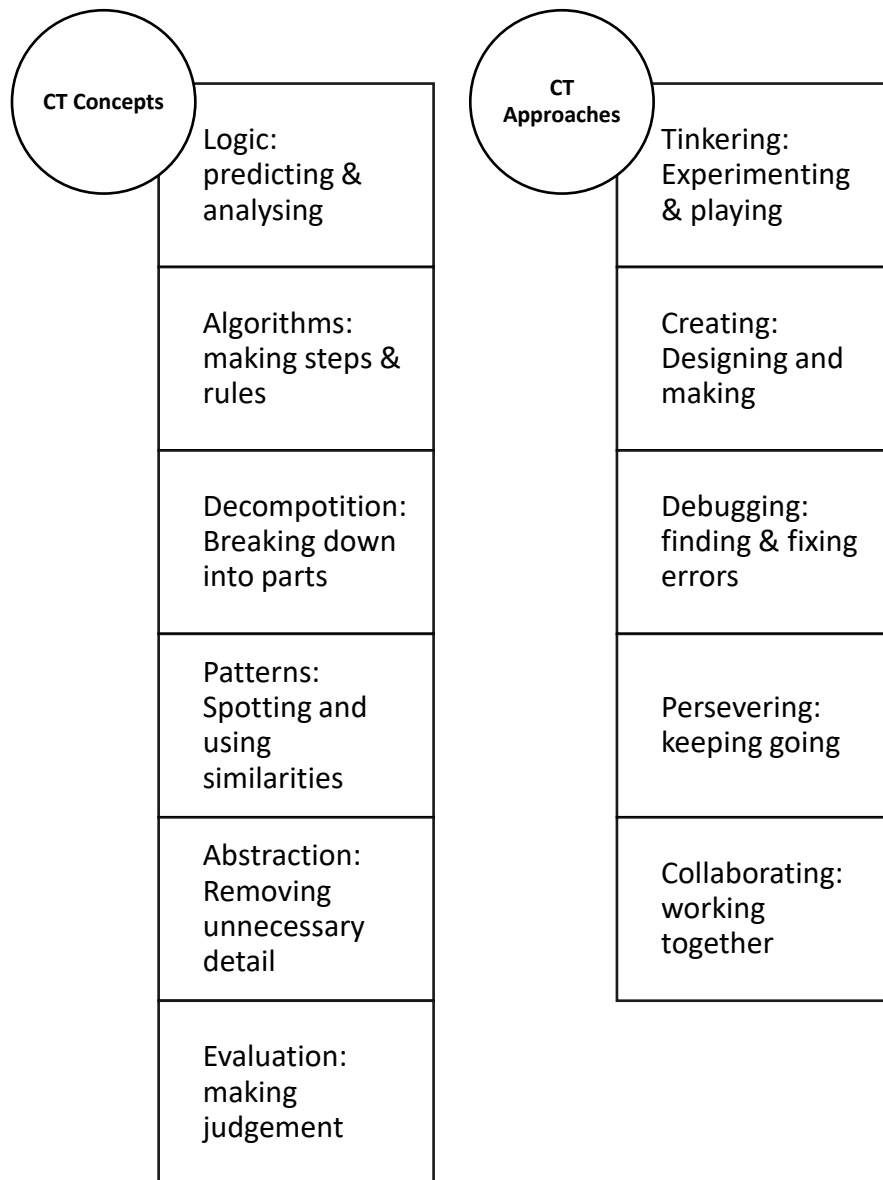


Figure 2-3. Computational Thinking Concepts and Approaches (Csizmadia et al., 2015).

Table 2-5. Computational Thinking Concepts (Csizmadia et al., 2015)

	Element	Description	In classroom
			<i>Examples of learner behavior that may be observed in the classroom</i>

Computational Thinking Concepts	Logic: the ability to think in terms of logical reasoning	the ability to analyze and control facts through thinking clearly and accurately to draw conclusions	
	Algorithms: the ability to think algorithmically	the ability of getting to a solution through a clear definition of the steps	<p>formulating instructions to achieve a desired effect</p> <p>using an appropriate notation to write code to represent the formulated instructions</p> <p>designing algorithmic solutions that take into account the abilities, limitations and desires of the people who will use them</p>
	Decomposition: the ability to think in terms of decomposition	the ability of splitting a whole into separate elements, thus reducing the level of difficulty in solving, understanding or designing	<p>breaking down artefacts into constituent parts to make them easier to work with</p> <p>breaking down a problem into simpler versions of the same problem that can be solved in the same way</p>
	Patterns: the ability to think in generalisations, identifying and making use of patterns	the ability of identifying patterns, similarities and connections, and exploiting those features	<p>identifying patterns and commonalities in artefacts.</p> <p>adapting solutions, or parts of solutions, so they apply to a whole class of similar problems</p> <p>transferring ideas and solutions from one</p>

			problem area to another.
	Abstraction: the ability to think in abstractions, choosing good representations	the ability of reducing unnecessary detail, so that a problem becomes easier or a concept simpler, without losing anything important.	<p>Choosing a way to represent an artefact, to allow it to be manipulated in useful ways</p> <p>hiding the full complexity of an artefact (hiding functional complexity)</p> <p>hiding complexity in data, for example by using data structures.</p> <p>Identifying relationships between abstractions.</p> <p>Filtering information when developing solutions.</p>
	Evaluation: the ability to think in terms of evaluation	ensuring that a solution, whether an algorithm, system, or process, is a good one: that it is fit for purpose	<p>assessing that an artefact is fit for purpose, is functional correct, is good enough, is easy for people to use and gives an appropriately positive experience when used</p> <p>designing and running test plans and interpreting the results (testing)</p> <p>comparing the performance of artefacts that do the same thing</p> <p>making trade-offs between conflicting demands</p>

			stepping through processes or algorithms/code step-by-step to work out what they do (dry run/tracing)
--	--	--	---

Table 2-6. Computational Thinking Techniques (Csizmadia et al.,2015)

	Element	Description
Computational Thinking Techniques	reflecting	making evaluations that have value
	coding	converting a plan into code and ensure that it produces the right result under any circumstances; debugging is the ability of evaluating, testing and verifying the outcome
	designing	creating representations of the design such as flowcharts, storyboards, pseudo-code, systems diagrams, etc. It involves further activities of decomposition, abstraction and algorithm design.
	analysing	breaking down into component parts (decomposition), reducing the unnecessary complexity (abstraction), identifying the processes (algorithms) and seeking commonalities or patterns (generalisation); using logical thinking both to better understand things and to evaluate them as fit for purpose
	applying	adoption of pre-existing solutions to meet the requirements of another context

2.3.5 International Society for Technology in Education (ISTE) and Computer Science Teacher Association (CSTA) (2011)

International Society for Technology in Education (ISTE) and Computer Science Teacher Association (CSTA) (2011) developed an operational definition that includes, the following elements:

- (a) formulating problems in a way that enables us to use a computer and other tools to help solve them,
- (b) logically organizing and analyzing data,
- (c) representing data through abstractions such as models and simulations,
- (d) automating solutions through algorithmic thinking (a series of ordered steps),
- (e) identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and
- (f) generalizing and transferring this problem-solving process to a wide variety of problems.

In addition to these elements (ISTE) and (CSTA) include the following attitudes to their operational definition:

- (a) confidence in dealing with complexity,
- (b) persistence in working with difficult problems,
- (c) tolerance for ambiguity,
- (d) the ability to deal with open ended problems; and
- (e) the ability to communicate and work with others to achieve a common goal or solution.

2.3.6 Kalelioglu, Gulbahar, & Kukul (2016)

Kalelioglu, Gulbahar, & Kukul (2016) develop a framework (Table 2-7) that describes CT skills, considering CT to be a problem-solving process.

Table 2-7. Framework for Computational Thinking as a Problem-Solving Process (Kalelioglu, Gulbahar, & Kukul, 2016)

Identify the problem	Gathering, representing and analysing data	Generate, select and plan solutions	Implement solutions	Assessing solutions and continue for improvement
Abstraction Decomposition	Data collection Data analysis Pattern recognition Conceptualising Data representation	Mathematical reasoning Building algorithms and procedures Parallelisation	Automation Modelling and simulations	Testing Debugging Generalisation

Each element of this framework is related to the process of solving a problem. For example, the problem must first be identified. Using subtraction and decomposition one can locate the structural elements of a problem. This is followed by data collection, representation and analysis. In the second phase, solutions that require mathematical reasoning and algorithmic reasoning must be designed. These solutions must be implemented using automation, modeling and simulations. Finally follows the evaluation of the implemented solutions using testing of the designed plans, debugging the code and generalization of the solutions.

2.3.7 Selby (2013)

Selby (2013) review the literature, analyzing the Computational Thinking terms previously proposed. She suggests Abstraction and Decomposition be at the heart of Computational Thinking and classifies the various terms found in the literature into four main categories: Thinking terms, Problem-solving terms, Computer Science terms and Imitation terms. After analyzing these terms, she proposes to exclude broad terms that are not well-defined and terms related to skills demonstrations, defining CT as a thought process that involves the following elements:

- (a) the ability to think in abstractions,
- (b) the ability to think in terms of decomposition,
- (c) the ability to think algorithmically,
- (d) the ability to think in terms of evaluations; and
- (e) the ability to think in generalizations.

2.3.8 Shute et al. (2017)

Shute et al. (2017) reviewed definitions and CT models and define CT as “the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts.” They propose an operational definition that underlines the relationship between Computational Thinking and problem solving and includes six main dimensions presented in Table 2-8.

Table 2-8. Computational Thinking components (Shute et al., 2017)

Facet	Description
Decomposition	Dissect a complex problem/system into manageable parts. The divided parts are not random pieces, but functional elements that collectively comprise the whole system/problem.
Abstraction	Extract the essence of a (complex) system. Abstraction has three subcategories: <ul style="list-style-type: none"> (a) Data collection and analysis: Collect the most relevant and important information from multiple sources and understand the relationships among multilayered datasets; (b) Pattern recognition: Identify patterns/rules underlying the data/information structure; (c) Modeling: Build models or simulations to represent how a system operates, and/or how a system will function in the future.

Algorithms	<p>Design logical and ordered instructions for rendering a solution to a problem. The instructions can be carried out by a human or computer. There are four sub-categories:</p> <p>(a) Algorithm design: Create a series of ordered steps to solve a problem;</p> <p>(b) Parallelism: Carry out a certain number of steps at the same time;</p> <p>(c) Efficiency: Design the fewest number of steps to solve a problem, removing redundant and unnecessary steps;</p> <p>(d) Automation: Automate the execution of the procedure when required to solve similar problems.</p>
Debugging	Detect and identify errors, and then fix the errors, when a solution does not work as it should.
Iteration	Repeat design processes to refine solutions, until the ideal result is achieved.
Generalization	Transfer CT skills to a wide range of situations/domains to solve problems effectively and efficiently.

2.3.9 Sondakh et al. (2020)

Sondakh et al. (2020) proposed a CT definition (Figure 2-4) based on the fuzzy Delphi method. Experts from Computer Science and Technology Industry participated in the study validating terms found in the literature. They describe the main components of CT as skills, attitudes and spiritual intelligence.

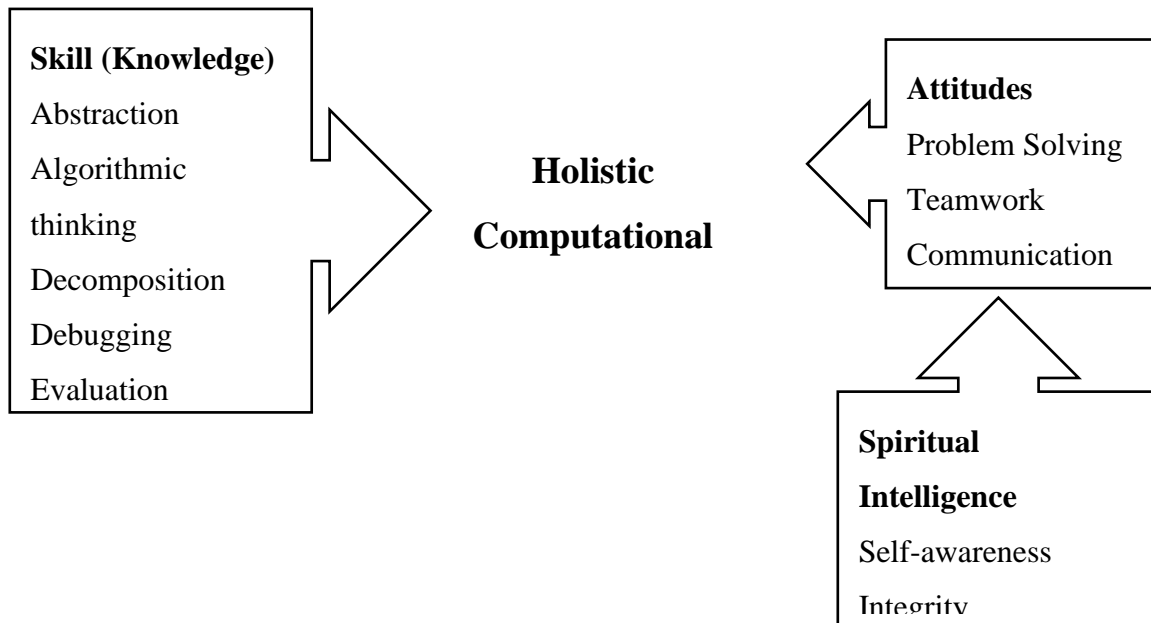


Figure 2-4. Definition of holistic Computational Thinking (Sondakh et al., 2020).

2.3.10 Weintrop et al. (2016)

Weintrop et al. (2016) proposed a definition (Figure 2-5) for Computational Thinking with an emphasis on mathematics and science. Their model consists of the following interrelated practices: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices. In addition, each category further consists of a subset of practices.

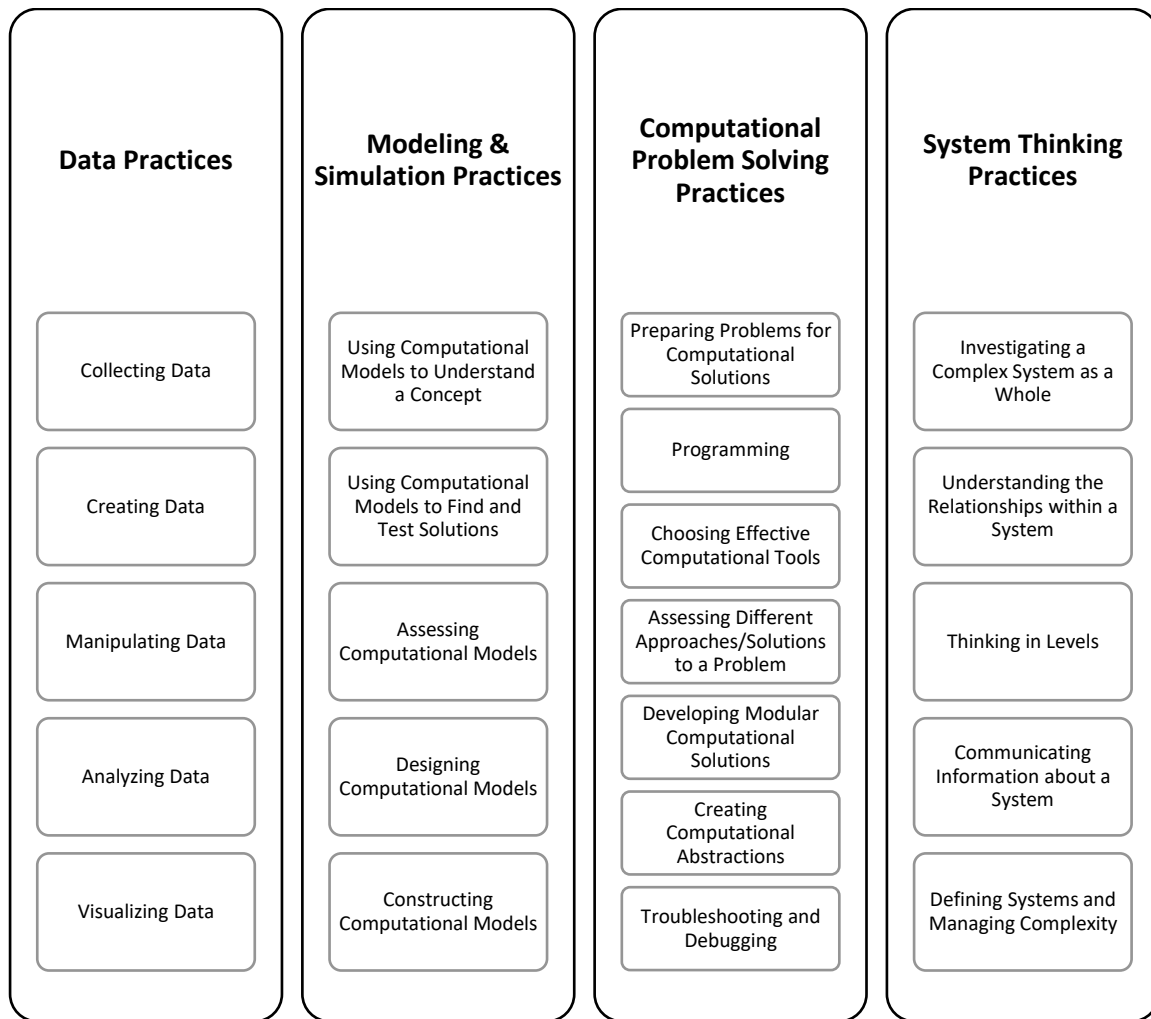


Figure 2-5. Computational Thinking in Mathematics and Science definition (Weintrop et al., 2016)

2.3.11 Computational Thinking elements

Each definition presented above contributes to the understanding and clarification of the Computational Thinking construct in relation to the elements of which it consists. This subsection presents a summary of CT elements (Table 2-9) described in some of the definitions presented in the previous sub-sections. We select to present the specific definitions as they are highly cited in the literature, cover an extensive period of time and are developed based on different approaches (e.g., systematic literature review, previous authors' studies, literature summary, meeting procedures).

Table 2-9. CT elements in CT operational definitions

Barr & Stephenson (2011)	Brennan & Resnick (2012)	Selby (2013)	Angeli et al. (2016)	Shute et al. (2017)
Abstraction Analysis and Model validation Simulation Data collection, analysis and representation	Abstracting and modularizing	Ability to think in abstractions	Abstraction	Abstraction Data collection and analysis Pattern recognition Modeling
Problem decomposition		Ability to think in terms of decomposition	Decomposition	Decomposition
Algorithms and procedures Control structures Parallelization Automation	Computational concepts (mapping to Scratch programming blocks such as sequences, loops etc.)	Ability to think algorithmically	Algorithms Sequencing Flow of control	Algorithms Algorithm design Parallelism Efficiency Automation
Testing and verification	Testing and debugging	Ability to think in terms of evaluations	Debugging	Debugging

		Ability to think in terms of generalizations	Generalization	Generalization
	Being incremental and iterative			Iteration
	Reusing and remixing			
	Expressing			
	Connecting			
	Questioning			

2.4 Computational Thinking Literature Reviews

Many of the above definitions come from studies that review the literature, investigating Computational Thinking terms. Despite the strong interest in defining Computational Thinking, a growing number of literature reviews focusing on the teaching and learning of Computational Thinking can be also found. Some of these reviews focus on a specific topic of CT domain, such as assessment, while others cover multiple topics. Reviews that cover multiple topics can be classified in three categories: a) studies aiming to develop a definition (e.g. Kalelioglu et al., 2016; Shute et al.; 2017) b) studies reviewing the literature to provide insights on teaching and learning CT (e.g. Grover & Pea 2013; Lye & Koh, 2014; Buitrago Flórez et al., 2017) and c) studies aiming to analyze CT research (e.g. Hsu et al., 2018). Despite all this work reviewing various aspects of CT through programming in K-12 education, a conceptual model of the domain is still missing. Table 2-10 presents recent Computational Thinking studies that review the literature

Table 2-10. Literature Reviews in CT domain

Review	Main Contribution	Scope	CT approach	Main focus on educational level	Studies included
(Grover & Pea, 2013)	Review CT definitions, the rationale for integrating CT into K-12 education, tools for CT development and assessment, and provide information on what CT entails and how is integrated in K-12 education.	General	Programming	K-12	Undefined
(Lye & Koh, 2014)	Review the trends of empirical research in the development of CT through programming in K-12 education such as programming environments, learning outcomes and approaches, and derive insights on K-12 curriculum.	General	Programming	K-12	Empirical higher education and K-12 articles
(Kalelioglu et al., 2016)	Review theoretical basis, definition, CT elements, population, type of research design, and develop	General	Programming and unplugged methods	K-12	Higher education and K-12 articles

	a framework that includes notion, scope and elements of CT.				
(Buitrago Flórez et al., 2017)	Review challenges faced by early programmers, programming languages and pedagogical tools, and provide an overview of how programming is being taught in K-12 and higher education.	General	Programming	K-12 and higher	Journal articles, reviews, proceedings, short communications, and governmental standards
(Shute et al., 2017)	Review CT definitions and characteristics, interventions, assessments and models, and develop a CT competency model.	General	Programming and other approaches	K-12	Conceptual papers and empirical studies
(T.-C. Hsu et al., 2018)	Review learning strategies, teaching instruments, programming languages and course types, and analyze the evolution of CT research.	General	Programming and other approaches	All educational levels	SCI and SSCI journal articles

(Ching, Hsu, & Baldwin, 2018)	Review the technologies used for developing CT in young learners.	Focused on technologies	Programming and other approaches	K-12	Undefined
(Da Cruz Alves, Gresse Von Wangenheim, & Hauck, 2019)	Review the automatic assessment tools used to analyze artifacts in order to assess CT skills.	Focused on automatic assessment	Programming	K-12	K-12 and higher education articles
(Zhang & Nouri, 2019)	Review the CT skills that can be obtained through Scratch in K-9 education and extend Brennan & Resnick's (2012) framework.	Focused on CT elements	Scratch programming	K-9	K-9 empirical studies

2.5 Summary

This Chapter presents definitions, related introductory concepts and previous work conducted in the field of Computational Thinking for a better understanding of the subject. Specifically, definitions and frameworks of Computational Thinking are presented and synthesized. In addition, literature reviews conducted in the field of Computational Thinking, are presented.

3 Methodology

The research was organized in the following four phases guided (Figure 3-1) by the respective research objectives presented in Section 1.1.

Phase 1. Developing a Conceptual Model of Computational Thinking through programming in K-12 education (CTPK-12).

Phase 2. Expanding the Computational Thinking through Programming in K-12 Education (CTPK-12) Conceptual Model for Higher Education.

Phase 3. Designing and evaluating of a Scaffolding Computational Thinking game to to be further used in the fourth phase of this dissertation.

Phase 4. Using the CTPK-12 model to design an empirical study to investigate certain instances of the Learning Strategies and Factors model's areas.

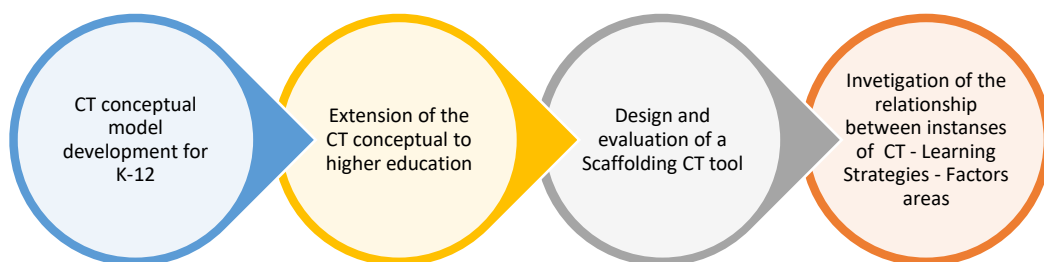


Figure 3-1. Method followed in this thesis

3.1 Phase 1. Developing a Conceptual Model of Computational Thinking through programming in K-12 education (CTPK-12)

In order to develop a Conceptual Model of Computational Thinking through programming in K-12 education we elicited the domain knowledge and we subsequently visualized this knowledge as proposed by Wand and Weber (2002). To gain the knowledge of the domain, we systematically reviewed the literature. To this end, we followed the Webster and Watson's (2002) method, widely used in conducting literature reviews. The method

involves a rigorous approach to the selection of cases to be included in the review and a concept-centric approach to the presentation of results. In addition, in order to further enhance the systematic selection of studies and to reduce subjectivity as much as possible, we applied the PRISMA statement (Moher, Liberati, Tetzlaff, & Altman, 2009). The steps of the method followed in this phase are further elaborated in Section 4.2.2.

3.2 Phase 2. Extending the Computational Thinking through Programming in K-12 Education (CTPK-12) Conceptual Model for Higher Education

This phase aims to extend the CT conceptual model to include higher education as well. For this purpose, we followed a systematic mapping method proposed by Petersen et al. (2008). They propose systematic mapping as a method that could contribute in research development by providing a structured type of research that has been conducted. The steps of the method followed in this phase are further elaborated in Section 5.2.2.

3.3 Phase 3. Designing and evaluating of a Scaffolding Computational Thinking game.

This phase aims to design and evaluate a Scaffolding Computational Thinking game. The decision to design a simple tool with scaffolding features instead of using a pre-existing tool was based on the need for customization. We based the design of the tool on the results of the literature review conducted in the previous phases. Specifically, we designed the tool in order to cover CT components included in Brennan and Resnicks' (2012) framework. The tool that includes features for scaffolding, also offers the ability to create log files for CT assessment. In order to evaluate the tool, we investigated how students perceive its effectiveness, paying particular attention to the scaffolding features. To do so, we collected data from students through a survey and analyzed their answers using descriptive statistics and thematic analysis. The research design of this phase is further elaborated in Section 6.2.

3.4 Phase 4. Investigating certain instances of the Learning Strategies and Factors CTPK-12 model's areas.

This phase aims to investigate certain instances of the Learning Strategies and Factors Areas presented in the CPTK-12 model. More specifically, this phase aims to investigate the effect of scaffolding programming games on middle school students' Computational Thinking acquisition. An additional goal is to investigate the effect of middle school students' attitudes towards programming in their Computational Thinking development. For this purpose, we designed and conducted an experimental study where students were randomly assigned to two groups. The students of the experimental group were introduced to Computational Thinking through the Scaffolding Computational Thinking tool, while the students of the control group were introduced using the same tool but without scaffolding features. Data were collected through tests, questionnaires and log files and were analyzed through descriptive and inferential statistics. The research design of this phase is further elaborated in Section 7.3.

4 Computational Thinking through programming in K-12 Education (CTPK-12) Conceptual Model

4.1 Introduction

The aim of this Chapter is to present one of the main theoretical results of this research, the Computational Thinking through programming in K-12 Education (CTPK-12) conceptual model.

The CTPK-12 could aid domain understanding and serve as a basis for future studies. It could also support researchers to focus on significant research gaps in their CT studies, having an up-to-date synthesis of the relevant literature. In addition, it could support the integration of CT into K-12 educational practices, providing evidence to teachers and policy-makers as well as bringing closer research, practice and policy.

The remainder of this Chapter is organised as follows: Section 4.2 presents the design of the study for the development of the CTPK-12 model. Section 4.3 presents the CTPK-12 model. Section 4.4 further discusses the CTPK-12 model areas and its potential uses. Section 4.5 presents the summary of the chapter.

4.2 Study design

4.2.1 Study goal and research questions

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

The research questions are:

RQ1. What are the areas of CT through programming in K-12 education domain?

RQ2. What are the sub-areas of each CT Area?

RQ3. How do CT Areas relate to each other?

4.2.2 Method

In order to develop a conceptual model for CT through programming in K-12 education we proceed to the following two steps proposed by (Wand & Weber, 2002): a) elicit the domain knowledge and b) visualize the domain knowledge. Figure 4-1 presents the study method in terms of steps conducted and relevant results. We apply the Webster and Watson’s (2002) systematic literature review approach for the elicitation of the domain knowledge (CT Areas and their relationships). This includes a structured approach to identifying sources and a concept-centric approach to presenting the results. We started by applying the PRISMA Statement (Moher et al., 2009) for the study selection phase. We then, proceed to the coding scheme identification phase, in which we identify the CT Areas that serve as a coding scheme for the data extraction phase. The data extraction phase aims to identify the sub-areas of each CT Area and the CT Areas’ relationships. The process concludes with the visualization of the data extraction phase results. The whole process evolved into iterative phases where searches led to new selected studies that were being analyzed, leading to revised CT areas, sub-areas and relationships. The steps followed in this study are further elaborated below.

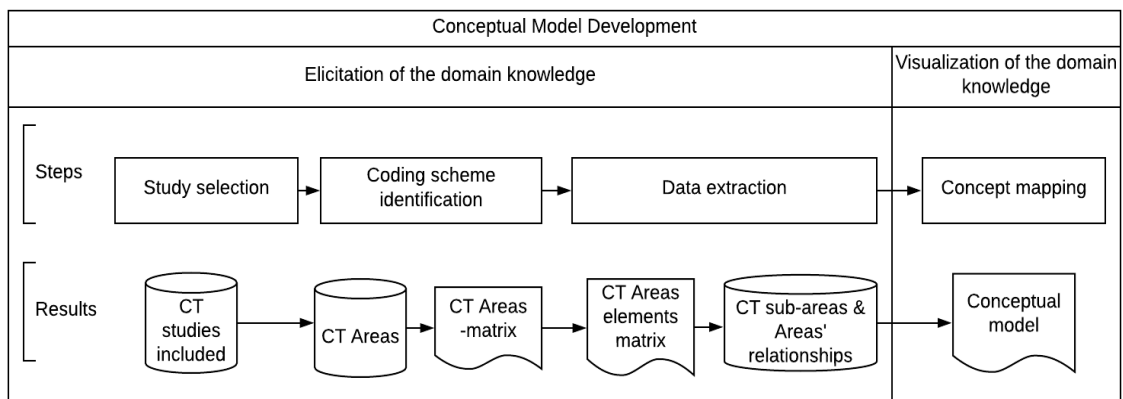


Figure 4-1. Method

4.2.2.1 Elicitation of the domain knowledge

Study selection

We carry out the study selection presented in (Figure 4-2), adapting the PRISMA Statement (Moher et al., 2009). Specifically, we adapt the PRISMA flow diagram (Figure

4-2) by placing additional records identified in included phase, as we identified these studies by examining the selected studies as proposed by Webster & Watson (2002).

The selection of studies included is a critical factor for the validity of the study. For this reason, the authors identified the search keywords and criteria together but worked individually to screen the studies and apply the inclusion and exclusion criteria. During this process a few conflicts emerged, which were solved through discussions until agreement was reached.

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

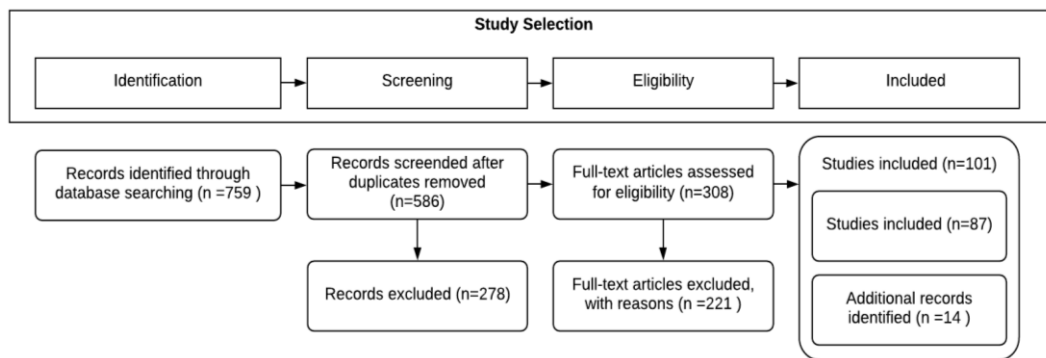


Figure 4-2. Process applied for study selection adapted by Moher et al. (2009)

Identification

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

Screening

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

Eligibility

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

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

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

Included

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

4.2.2.1.1 Coding scheme identification

To determine the areas of CT through programming in K-12 domain that serve as our coding scheme, we apply conventional content analysis. Conventional content analysis is suggested when existing theory is limited and does not involve a predefined coding scheme but one that derives from text analysis (Hsieh & Shannon, 2005). We choose conventional content analysis because of the lack of a conceptual model describing the domain. Initially, we read all full-text articles in order to approach the domain as a whole. Then we carefully read each article and highlight keywords that imply a concept/area. Keywords are combined together, providing categories of the coding scheme. For example, keywords “assessing the development of Computational Thinking”, “assessment” (Brennan & Resnick, 2012), “assess and evaluate”, “assessment” (Zhong et al., 2016) are grouped and eventually led to adding “Assessment Area” in the coding scheme. Subsequently, we sort the studies in these categories. During this phase the coding scheme evolves by adding new categories or merging and splitting existing ones. The phase leads to the identification of the final categories, which from now on will be referred to as CT Areas and serve as the coding scheme and as the concepts of the conceptual model.

Consequently, we compile a concept-matrix or CT Area-matrix, which is a matrix listing the CT Areas where each article contributes. In this way we transit from an author-centric to a concept-centric approach, as suggested by Webster and Watson (2002) (Table 4-1).

Table 4-1. Approaches to Literature Reviews adopted from Webster and Watson (2002)

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

4.2.2.1.2 Data extraction

During this phase, we sort the selected studies into the coding scheme. In this respect, we use a table for each CT Area. When we insert a study into the table, we also record the area’s elements that appear in the study (Figure 4-3). Subsequently, we compare every element with all other elements. The elements with clear match with other elements

constitute a sub-area. For example, in Assessment Area, “project analysis” (Brennan & Resnick, 2012) and “examination of artifacts for CT patterns” (Denner et al., 2012) are included in the “Artifact analysis” sub-area. Sub-areas consisting of only one element and low-frequency (<2 studies) sub-areas, are not included in the model.

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

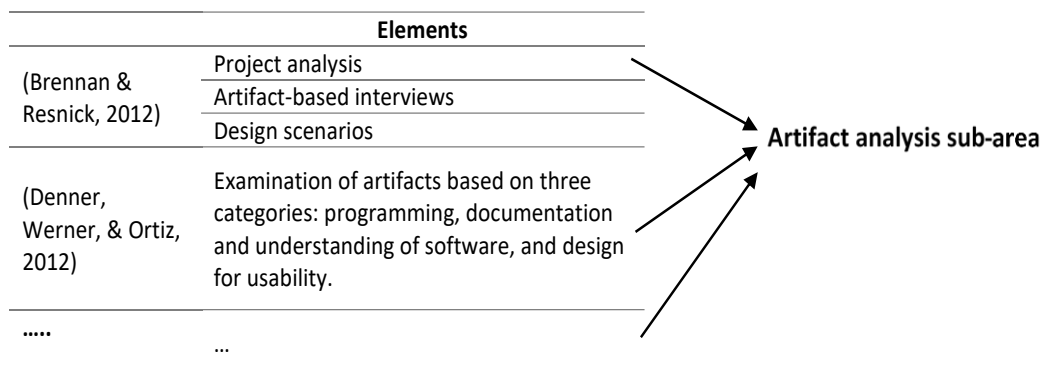


Figure 4-3. Example of elements recording and sub-areas identification

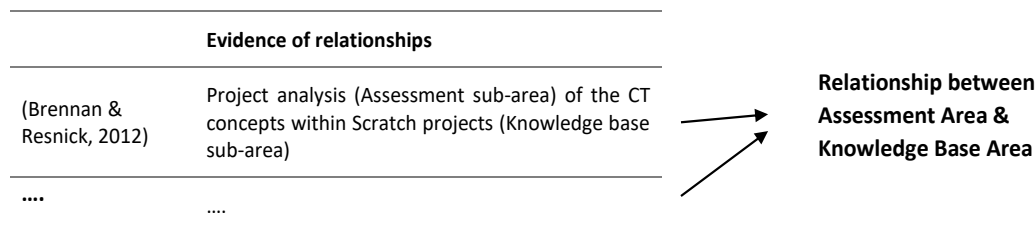


Figure 4-4. Example of evidence recording and relationship identification

4.2.2.2 Visualization of the domain knowledge

4.2.2.2.1 Concept mapping

In this step, we use concept mapping as proposed by Siau & Tan, (2005) for visualizing the concepts (CT Areas) and relationships of the domain, the identification of which is described in section 4.2.2.1.1. We create a visualization of the conceptual model depicting CT Areas as nodes. At each node, we note the sub-areas of each CT Area, identified in the

previous phase. Finally, we depict the relationships between CT Areas as links. We then place a label to each link to explain the relationship.

4.2.3 Study limitations

We acknowledge that this study has a number of limitations. First, the proposed model is based on the analysis of studies written in English. Second, searches for studies were conducted in only two scientific databases, namely Web of Science and Scopus. Third, searches included only articles published in journals. Although, we eventually included some conference papers and gray literature identified through manual inspection of the references of the selected studies, still the majority of the selected literature includes journal articles. Fourth, searches were conducted with a time constraint of 2006 onwards. Thus, the model is based exclusively on the research conducted since 2006 and not on the initial stages of CT research. Fifth, non-inclusion of studies on the basis of quality criteria prevents the presentation of all conducted research. Finally, subjectivity combined with the small number of authors (only two) constitutes an additional limitation of the study. Although we applied a systematic method (presented in Section 4.2.2) we had to make subjective choices regarding e.g., grouping the elements, defining the relationships based on the recorded evidence, naming the CT Areas and sub-areas, and defining exclusion criteria for selecting sub-areas that are finally included in the model.

4.3 The CTPK-12 model

The proposed Computational Thinking through Programming in K-12 education (CPTK-12) conceptual model (Figure 4-5) is based on the extracted CT Areas and their relationships presented in detail in Sections 4.3.1 and 4.3.2 respectively.

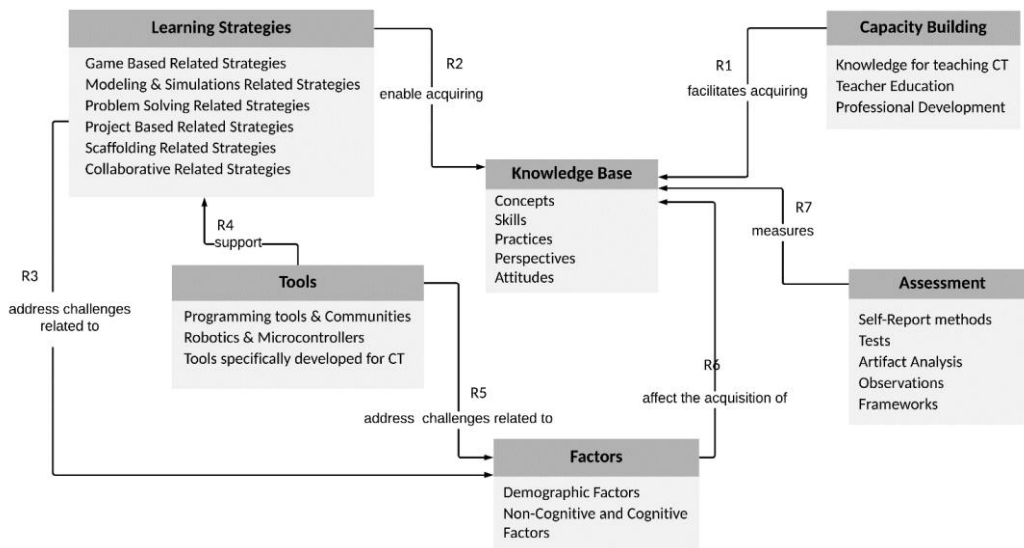


Figure 4-5. Computational Thinking through Programming in K-12 education (CTPK-12) model

The analysis of the 101 studies during the coding scheme identification phase resulted in the determination of six CT Areas finally included in the model (Table 4-2). CT studies attempt to address the challenges of CT through programming in K-12 education domain by focusing on these areas that repeatedly appear in the selected studies.

Table 4-2. CT Areas

Knowledge Base	CT measurable elements and their classification.
Assessment	Assessment methods and frameworks for measuring CT through programming in K-12 education.
Learning Strategies	Learning strategies leveraged to enhance students' CT learning through programming in K-12 education.
Factors	Factors related to CT through programming acquisition in K-12 education.
Tools	Tools that are used or specifically developed for teaching and learning CT through programming in K-12 education.

Capacity Building

Capacity building needed for teaching CT through programming in K-12 competently.

The percentage of studies by CT Areas to which they contribute is depicted in Figure 4-6. We categorize the studies into two groups 2006-2014 and 2015-2019. As shown in Figure 4-6, Assessment and Tools are the two most popular areas that gather the greatest interest of researchers in both periods. Assessment Area is coming first across the two timelines (27.9% in period 2006-2014, 25.6% in period 2015-2019) followed by Tools Area (20.9% in both periods). During period 2006-2014 Knowledge Base Area is coming third (18.6%) while in period 2015-2019 the percentages of studies aimed at defining CT fall to 8.5% placing the area as the one with the least interest. On the contrary, the percentage of studies that focus on Learning Strategies increases from 9.3% during period 2006-2014 to 17.1% during period 2015-2019, placing Learning Strategies in the third place of researchers' interest in the selected studies. Respectively for the Capacity Building Area the percentage of studies that focus on this area increases from 9.3% during period 2006-2014 to 14.7% during period 2015-2019, placing Capacity Building in the fourth place of interest followed by Factors. These results indicate that as the field matures efforts still focus on assessment and tools but the focus shifts beyond the definition of CT on more tangible issues such as Learning Strategies, Capacity Building and Factors.

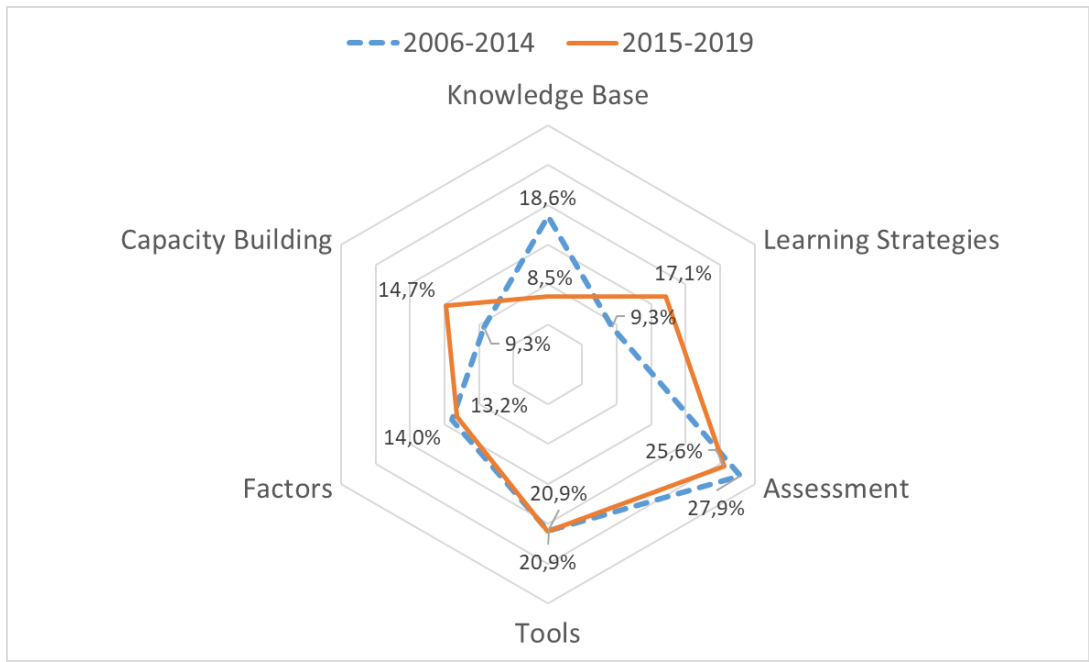


Figure 4-6. Percentage of studies by CT Areas to which they contribute in the periods 2006-2014 and 2015-2019. References to 2019 actually refer to period January 2019 to October 2019

The CTPK-12 depicts the areas of Computational Thinking through programming in K-12 education (CT Areas) and the dominant relations (Table 4-3) between CT Areas as they emerge from the selected studies.

Table 4-3. CT Areas’ relationships

Capacity Building	R1. Supports teachers to facilitate students understand and acquire CT.
Learning Strategies	R2. Enable students understand and acquire CT. R3. Increase the motivational levels of underrepresented students, thereby broadening CT participation and addressing underrepresentation due to socio-economic, cultural and gender differences.
Tools	R4. Allow students to acquire CT through supporting learning strategies.

	R5. Address the challenges encountered in learning programming and reinforce underrepresented students' motivation.
Factors	R6. Affect the acquisition of CT.
Assessment	R7. Measures CT and provides a means for deep understanding of students' learning.

4.3.1 CT Areas

4.3.1.1 Knowledge Base Area

Knowledge Base Area is at the core of the domain. 57 of the 101 studies are included in this CT Area. Researchers in these studies either propose a framework or a definition to identify and classify measurable elements of CT, or simply assess CT elements in order to assess CT. Based on CT frameworks we examined CT elements in the selected studies. We classify Knowledge Base Area in five sub-areas: concepts, skills, practices, perspectives and attitudes (Table 4-4). Figure 4-7 presents the number of studies by CT element.

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

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

The analysis of the reviewed studies reveals the following CT practices according to Brennan & Resnick’s (2012) framework: Testing and Debugging, Remixing and

Reusing code, Being incremental and iterative, Abstracting and Modularizing. In addition, elements such as Design for usability, Code organization and documentation, and Programming efficiency proposed by Denner et al. (2012) as key competences for engaging in CT are also evident.

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

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

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

Table 4-4. Knowledge Base sub-areas

CT elements classification	Description	CT frameworks
Concepts	Concepts (programming elements) encountered during programming.	Brennan & Resnick (2012)

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

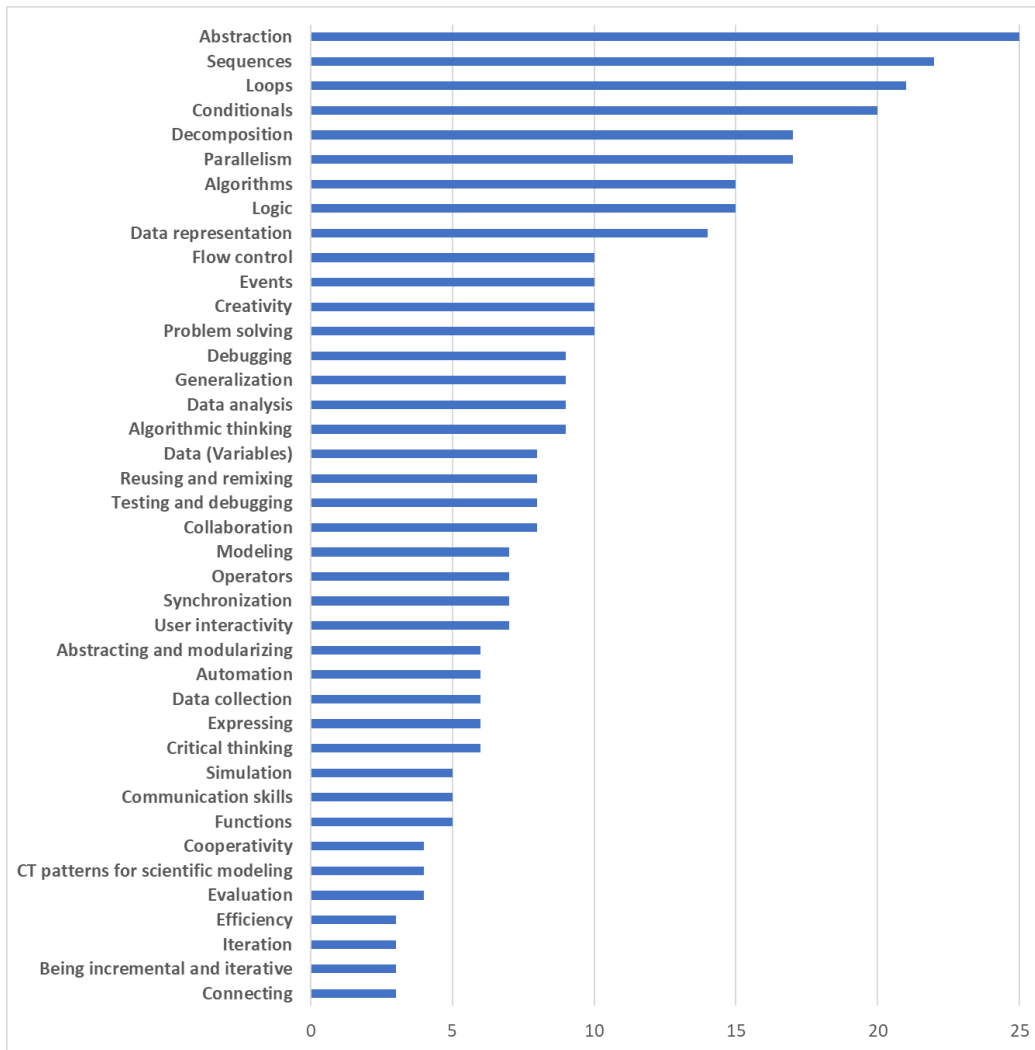


Figure 4-7. Number of studies by CT element appearing more than twice in the examined studies

4.3.1.2 Assessment Area

CT assessment is examined in 53 studies. Researchers in the examined studies develop and validate assessment methods, propose frameworks or measure students' CT in order to achieve deep understanding of students' learning (Fronza et al., 2017) through various assessment methods. We classify Assessment Area into five sub-areas: Self-report methods; Tests; Artifact Analysis; Observations; and Frameworks. Tests, Artifact Analysis and Observations measure directly CT, in contrary with self-report methods that measure CT indirectly through recording self-reflection. Table 4-5 presents the classification of Assessment.

CT assessment methods in the examined studies are mainly based on the specific content of each study. However, there are some efforts to develop general assessment methods. These efforts include development and validation of tests (Chen et al., 2017; Román-González et al., 2017), self-report scales (Kong, Chiu, & Lai, 2018; Korkmaz et al., 2017; Kukul & Karataş, 2019; Yağcı, 2019) for general use in CT assessment and automatic artifact analysis instruments (Moreno León et al., 2015). Artifact analysis involves examining students' programs to detect evidence of CT. Automatic artifact analysis allows teachers and researchers to focus on assessment methods such as observations and interviews to gain a complete picture of students' understanding (Da Cruz Alves et al., 2019).

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

Table 4-5. Assessment sub-areas

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

4.3.1.3 Learning Strategies Area

Learning strategies are mentioned in 37 studies. We classify the most common learning strategies in six sub-areas: Game Based Related Strategies, Modeling & Simulations Based Related Strategies, Problem Solving Related Strategies, Project Based Related Strategies, Scaffolding Related Strategies and Collaborative Related Strategies (Table 4-6). Scaffolding Related Strategies are classified as a separate sub-area, as they are particularly emphasized in the selected studies. Other strategies involve hands-on, aesthetic design through media design, storytelling and guided-discovery. Figure 4-8 presents the number of CT studies by most common strategies.

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

Table 4-6. Learning strategies sub-areas

		Studies
Game Based Related Strategies	Game Based Related Strategies involve game design and digital/video game development, programming games and any strategy that exploits games and programming.	S4,S25,S26,S35, S36,S46,S48,S53, S60,S72,S89, S100
Modeling & Simulations Based Related Strategies	Modeling & Simulations Based Related Strategies involve designing of scientific models and simulations through strategies such as scientific inquiry and learning by design.	S2,S11,S28,S35, S72,S81

Problem Solving Related Strategies	Problem Solving Related Strategies involve Problem Based Learning and problem-solving learning strategies in general.	S5,S39,S51
Project Based Related Strategies	Project Based Related Strategies involve the engagement with authentic projects set around real challenges and problems.	S32,S53,S69,S70,S79
Scaffolding Related Strategies	Scaffolding Related Strategies involve strategies that offer support to students as they learn, including instructional scaffolding, support/guidance, and adaptive, peer-, resource- scaffolding.	S6,S11,S13,S17,S26,S36,S39,S45,S70,S72,S81,S93
Collaborative Related Strategies	Collaborative Related Strategies involve strategies where students actively interact during the learning process including collaborative learning, teamwork, pair programming and strategies based on student's collaboration.	S6,S30,S33,S45,S48,S70

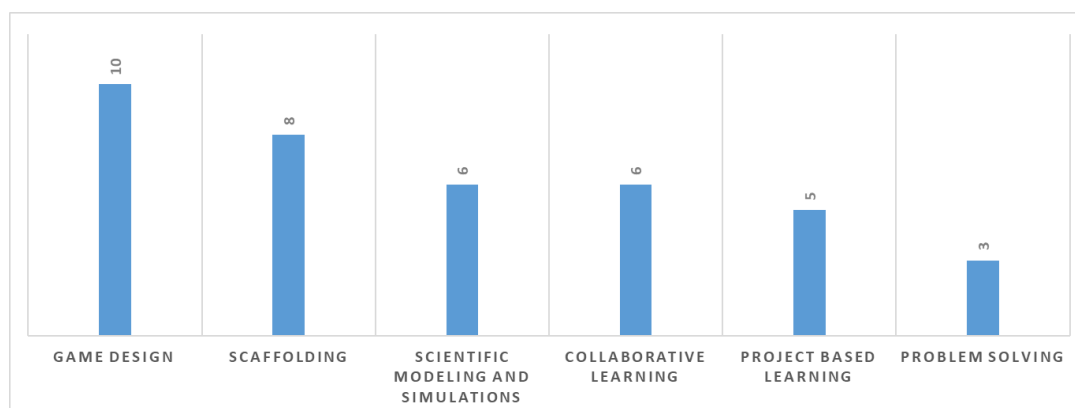


Figure 4-8. Number of studies by the most common learning strategies

4.3.1.4 Factors Area

CT-related factors are discussed in 22 studies. We classify Factors Area in two sub-areas: Demographic factors and Cognitive & non-cognitive factors presented in Table 4-7. Demographic factors have the strongest presence in the selected studies with gender being discussed in 17 and grade level in 7 out of 22 examined studies.

Several studies investigate the relationship between CT and grade level. Some of them (Atmatzidou & Demetriadis, 2016; Werner, Denner, Campe, & Kawamoto, 2012) conclude that CT acquisition is not grade-related (or age-related). Several other studies conclude that there is a significant relationship. However, their results on the type of this relationship are contradictory. On the one hand, some studies conclude that there is a positive relationship between grade level or age and CT. More specifically, Román-

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

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

Creativity appears in the selected studies in the light of two different perspectives. Several studies (Allsop, 2019; Kim & Kim, 2016; Korkmaz et al., 2017; Yağcı, 2019; Zhong et al., 2016) place creativity in the core of CT along with other elements. However,

other studies approach creativity as a separate construct and examine its relationship to CT. Teachers who participated in Nouri, Zhang, Mannila, & Norén (2019) reported creativity as one of the skills occurred during CT learning. Kim & Kim (2016) found that students' creativity was improved after they participated in their CT intervention. On the contrary, HersHKovitz et al. (2019) found no relationship between CT and creativity. However, they suggest that this may relate to specific features of the learning platform used.

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

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

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

Table 4-7. Factors sub-areas

		Studies
Demographic factors	Grade level, gender, socio-economic and cultural background	S4,S6,S22,S29,S30,S31,S43,S45,S49,S53,S55,S56,S70,S72,S76,S77,S90
Non-Cognitive and Cognitive factors	Personal traits, attitudes and motivations such as aspects of personality, creativity, self-efficacy, persistence, attitudes toward programming and	S29,S30,S42,S46,S55,S66,S76,S77,S81,S90, S93

attitudes toward collaboration; academic performance, challenges in learning programming

Factors that involve cognitive functions and mental abilities such as verbal, spatial, reasoning & numerical ability and problem-solving ability

4.3.1.5 Tools Area

Researchers in 47 studies use or develop tools for CT teaching and learning. We classify tools leveraged for teaching and learning CT through programming in K-12 education in three sub-areas: programming tools & communities, robotics & microcontrollers, and tools specifically developed for CT. Table 4-8 presents the classification of tools. Figure 4-9 presents the number of studies by tool.

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

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

Several studies develop tools in order to support a CT theoretical framework or curriculum. Most of the developed tools are visual programming tools and involve game play (e.g., Clark & Sengupta, 2019; Weintrop, Holbert, Horn, & Wilensky, 2016) and/or

modeling (e.g., Basu, Biswas, & Kinnebrew, 2017; Clark & Sengupta, 2019; Kynigos & Grizioti, 2018; Sengupta et al., 2013).

Table 4-8. Tools sub-areas

		Studies
Programming tools & Communities	Visual & text programming tools. Communities that provide users with the opportunity to interact with other programmers.	S2,S4,S5,S10,S15,21,S26,S30,S32,S33,S35,S36,S37,S39,S42,S44,S45,S46,S48,S49,S53,S54,S58,S60,S63,S70,S71,S72,S75,S79,S86,S94,S101
Robotics & Microcontrollers	Programmable robot constructs including educational robotics kits, physical & virtual robots. Automations, control devices, interactive physical systems.	S6,S12,S13,S17,S18,S19,S30,S60,S93
Tools specifically developed for CT	Tools developed to support a CT theoretical framework or curriculum.	S11,S21,S47,S59,S81,S89, S93, S100

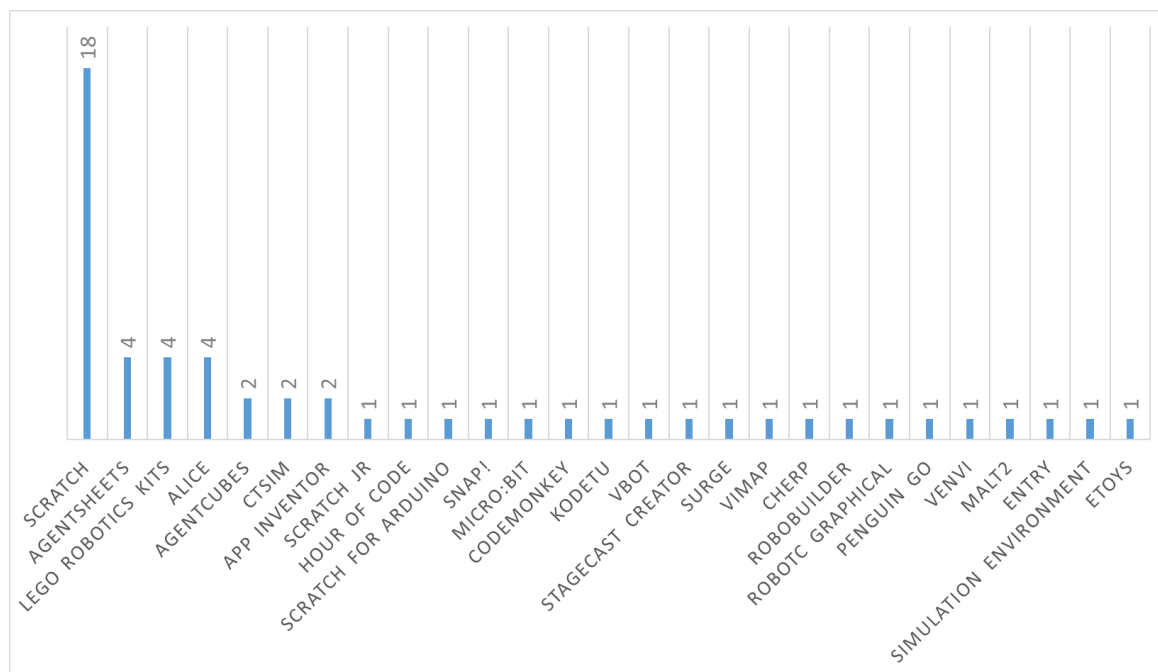


Figure 4-9. Number of studies by tool

4.3.1.6 Capacity Building Area

Providing guidance and support to teachers is discussed in 19 studies. We classify Capacity Building Area in three sub-areas: Knowledge for teaching CT, Teacher Education and Professional Development (Table 4-9).

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

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

Professional Development aims to support teachers in understanding and integrating CT into their practices (Alfayez & Lambert, 2019; Bower, Wood, Lai, Howe, & Lister, 2017). Hickmott & Prieto-Rodriguez (2018) propose that Professional

Development should (a) provide activities relevant to both CT tools and CT learning strategies; (b) include both step-by-step exercises and self-directed projects; (c) take into account teachers' prior knowledge; (d) provide resources that can be directly integrated into teaching practices; and (e) assess teachers' knowledge acquisition through direct assessment methods. Kale et al. (2018) argue that when Professional Development focuses on the application of CT in different domains and problem solving, it allows teachers to recognize the importance of CT and integrate the knowledge gained into their teaching. Ongoing professional development that involves workshops, embedded coaching, administrative support, co-planning lessons and co-teaching, could also provide in-service teachers with valuable assistance and thereby expanding their participation in CT (Israel, Pearson, Tapia, Wherfel, & Reese, 2015).

Table 4-9. Capacity Building sub-areas

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

4.3.2 CT Areas Relationships

CT Areas Relationships are depicted as arrows between the CT Areas in Figure 4-5 and described in Table 4-3. R6 and R7 model's relationships could be considered plausible and are widely reflected in the studies included in the respective CT Areas described in section (4.3.1). The same is true for R1, while this relationship is not widely tested empirically in the selected studies. The remaining relationships are further elaborated in this section.

R2. Several studies attribute the success of the proposed interventions to the applied strategies. Grover et al. (2015) place particular emphasis on the pedagogical design of their strategy, which eventually led to the students' understanding of CT concepts (algorithmic constructs). Repenning et al. (2015) also found that Scalable Game Design strategy that

involves game design, simulations and scaffolding allowed students to develop CT skills, highlighting the important role of pedagogy in the strategy. Sáez-López, Román-González, & Vázquez-Cano (2016) implemented an active pedagogical approach, concluding that primary school students who participated in their study improved their CT levels in regard to CT concepts, logic and CT practices. In addition, there are also findings that support the assumption that learning strategies such as Game Design (Garneli & Choriantopoulos, 2019), Project Based Learning enhanced with software agile methods (Fronza et al., 2017) and Modeling & Simulations (Garneli & Choriantopoulos, 2018) enable the acquisition of CT.

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

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

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

R5. There is also evidence that tools enhance underrepresented students' engagement in programming and CS. In a study by Kim & Kim (2016), participating elementary female students reduced their negative attitudes towards software education after following a CT course and designing games in App Inventor.

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

4.4 Discussion

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

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

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

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

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

Tools in the reviewed studies provide environments and communities where students are engaged with programming constructs and practices. Most of them share the common feature of visual programming. Scratch is the most commonly used tool and is usually used for game and media design. This is likely due to the combination of the following reasons: a) Scratch is proposed as a tool to support CT development by its designers (Resnick et al., 2009), b) Brennan & Resnick's (2012) framework in which CT elements are defined in relation with Scratch, facilitates researchers to use Scratch in their studies and c) the assessment of CT through projects developed in Scratch is facilitated by automatic assessment methods such as Dr. Scratch (Moreno León et al., 2015).

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

Capacity Building has gained attention especially after 2015. Teacher education, professional development and the knowledge that teachers need in order to teach CT are the main issues discussed in the selected studies. Many of these studies are surveys that

examine the challenges faced by teachers. Other studies propose frameworks or discuss professional development and teacher education interventions.

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

In addition, the CTPK-12 model could serve as a basis for hypothesized research models that establish a direct link between theory and statistical estimations. An example is presented in (Figure 4-10) where research hypothesis is developed between some CT Areas of the model. Research hypothesis in the specific example includes H1 (Between Learning Strategies Area and Knowledge Base Area): Game design enables the acquisition of CT skills. H2 (Between Learning Strategies Area and Factors Area): Game design motivates female students, addressing gender differences. H3 (Between Tools and Learning Strategies): Scratch provides opportunities for game development, supporting game design. H4 (Between Tools and Factors): Scratch motivates female students, addressing gender differences. H5 (Between Factors and Knowledge Base): Female and male students acquire a different level of CT skills.

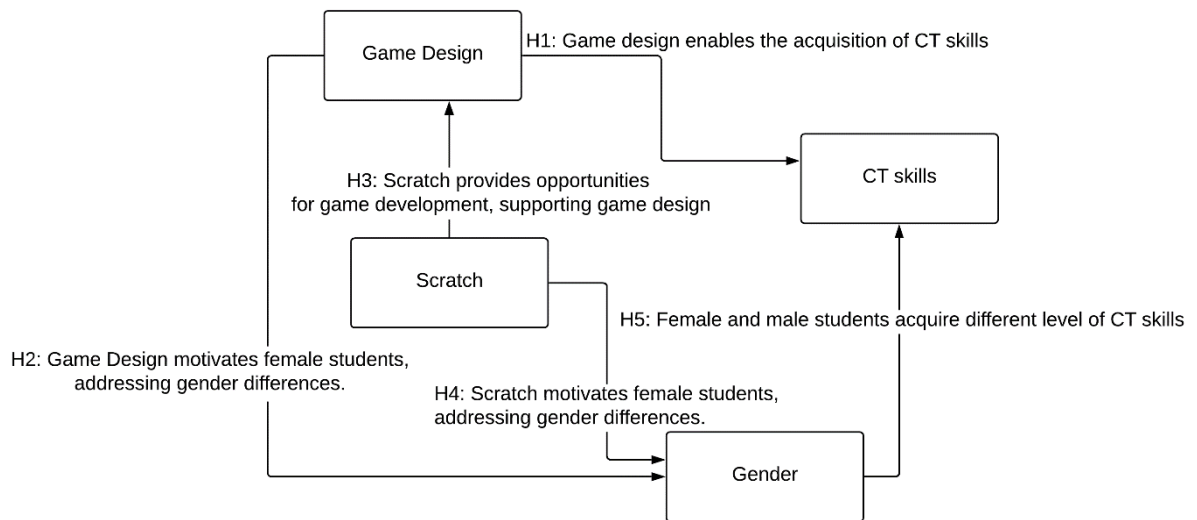


Figure 4-10. Example of a hypothesized research model based on CTPK-12 model

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

In addition, the CTPK-12 conceptual model could be combined with models for CT activities such as the scope of autonomy model (Carlborg et al., 2019) and the constructionism matrix (Csizmadia, Standl, & Waite, 2019). The CTPK-12 model could be used as a guide to designing teachers' lessons, providing them with evidence-based results and detailed information on CT through programming in K-12 education and facilitating them to integrate CT into their educational practices. The models' areas and their relationships could be taken into account during designing of curricula as well as CT teaching and learning process to improve effectiveness. In addition, CTPK-12 model could inform policy makers on their decision-making regarding CT and integration into K-12 education. It should be noted that the application of the CTPK-12 model in practice should take into account the settings under which CT will be incorporated. These settings include

parameters such as course type (optional or compulsory) or whether CT will be employed into other courses in the curriculum or as a separate course. Further elaboration of these settings is outside the scope of this study. Figure 4-11 presents the possible application of CTPK-12 model in educational practice.

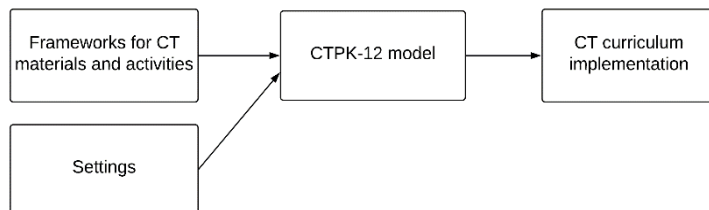


Figure 4-11. CTPK-12 model application in K-12 educational practice

4.5 Summary

This chapter presents the method and results of the first phase of this dissertation that involves investigating and analysing the literature in order to elicit the areas of Computational Thinking domain and their relationships. The purpose of this phase was to develop a conceptual model based on a systematic literature review that maps the CT through programming in K-12 education domain. The proposed Computational Thinking through Programming in K-12 education (CTPK-12) conceptual model emerges from the synthesis of 101 studies and the identification of CT Areas. The proposed model consists of six CT Areas (namely Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building) and their relationships.

5 Extending the CTPK-12 model for higher education

5.1 Introduction

In the previous Chapter we present the development of the (CTPK-12) Computational Thinking through Programming in K-12 Education conceptual model. We thoroughly analyse the concepts (CT Areas) and relationships of the model. In this Chapter, we proceed to extend the proposed CTPK-12 model to include higher education to develop a holistic model covering CT teaching and learning from early years until graduation.

The remainder of this Chapter is organised as follows: Section 5.2 presents the design of the study followed for the extension of the CTPK-12 model. Section 5.3 presents an overview of Computational Thinking through programming studies in higher education. Section 5.4 presents the CTPHE model which is the extension of the CTPK-12 for higher education. Section 5.5 further discusses the CTPHE model areas. Section 5.6 presents a summary of the chapter.

5.2 Study design

5.2.1 *Study goal*

The study goal is to expand the Computational Thinking through Programming in K-12 education (CTPK-12) Conceptual model for higher education.

5.2.2 *Method*

In order to achieve the study goal, we apply a Systematic Mapping Study based on Petersen's et al. (2008) methodology. This includes the following adapted steps.

Step1. Definition of research questions: Definition of research questions based on the study goal (Section 5.2.1)

Step2. Conduct search for primary studies: Conducting a structured search based on relevant search strings on scientific databases (Section 5.2.2).

Step3. Screening of Studies: Applying exclusion and inclusion criteria (Section 5.2.3).

Step4. Classification scheme Identification: Definition of the classification scheme (Section 5.2.4).

Step5. Data Extraction and mapping process: Shorting the studies into the classification scheme and provide visualizations of the results. Figure 5-1 presents the study method in terms of steps conducted and relevant outcomes.

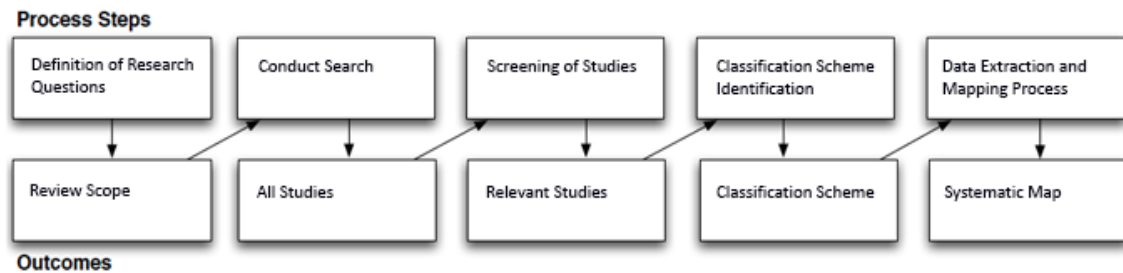


Figure 5-1. Systematic mapping process, adapted from Petersen et al. (2008)

5.2.2.1 Definition of Research Questions

The research questions are the following:

RQ1. What are the areas and sub-areas of teaching and learning CT through programming in higher education?

RQ2. How do these areas evolve over the years and how do they apply to various branches?

5.2.2.2 Conduct search for primary studies

We structured the search string driven by the research study goal. Specifically, we used the search string TITLE-ABS-KEY (“computational thinking”) AND PUBYEAR > 2005 AND (LIMIT-TO (DOCTYPE , “ar”) OR LIMIT-TO (DOCTYPE , “r”)) AND (LIMIT-TO (LANGUAGE, “English”)) in Scopus database and TITLE: (“computational thinking”) Refined by: DOCUMENT TYPES: (ARTICLE OR REVIEW) AND LANGUAGES: (ENGLISH) Timespan: 2006-2020. Indexes: SCI-EXPANDED, SSCI, A&HCI, ESCI in Web of Science database. Searches include articles published between January 2006 and December 2020. Searches took place on January 2021 and resulted in 993 studies, 707 articles in Scopus database and 286 in Web of Science database.

5.2.2.3 *Screening of studies*

During this step we removed 249 duplicates and studies that were not fully available. Subsequently, we applied inclusion and exclusion criteria to exclude studies that were not relevant to answering the research questions. Table 5-1 presents the exclusion and inclusion criteria defined. Finally, we included 39 primary studies and 2 additional primary studies that we identified through backward (reviewing citations) and forward searching. Appendix B present the total of 41 studies included.

Table 5-1. Inclusion and exclusion criteria

Inclusion Criteria	Exclusion Criteria
Empirical CT studies in which participants are undergraduate students, postgraduate students and academic staff.	Studies which discuss/apply CT through other means than programming.
Empirical CT studies that focus on CT through programming.	

5.2.2.4 *Classification Scheme Identification*

We use as base for the classification scheme the areas of the CTPK-12 model presented in Chapter 4. Each Area of the model corresponds to one category in the classification scheme. Petersen et al. (2008) propose the extraction of the classification scheme based on keywording of abstracts of the selected studies. For this purpose, we read all the abstracts of the selected articles and wrote down keywords. Each keyword was assigned to one of the classification scheme categories in order to determine if there were any additional categories that could be included in the classification scheme.

5.2.2.5 *Data extraction and mapping process*

In this step we classify the selected primary studies into the classification scheme. According to Petersen et al. (2008) the classification scheme evolves while data extraction is performed. When sorting the selected primary studies into the categories, new sub-categories appear, while others remain unused. We used an Excel table per category to

document the different instances of sub-categories in each primary study and the evolution of the classification scheme. When listing a primary study into a particular category and sub-category, we provide a brief rationale for why the study should be located in that particular category/sub-category. The final tables show the distribution of primary studies into sub-categories and calculate the relevant frequencies. The analysis of the results focuses on comparing frequencies between different time periods and different targeted groups. This allows us to identify the categories and sub-categories highlighted in CT through programming in higher education research and therefore understand its evolution and application.

5.2.3 Study Limitations

We acknowledge that this study has some limitations. First, the study includes only studies written in English. Second, searches were conducted in only two scientific databases, namely Web of Science and Scopus. Third, searches were conducted with a time constraint of 2006 onwards. Thus, the study maps the research conducted since 2006 and not on the initial stages of CT research. Finally, the small number of authors (only two) combined with subjectivity constitutes an additional limitation of the study. Although we applied a systematic mapping method, we had to make subjective choices regarding the evolution of the classification scheme.

5.3 Overview of Computational Thinking through programming studies in higher education.

5.3.1 Studies by year

The distribution of studies by year (Figure 5-2) reveals an upward trend in the number of studies. This is particularly true from 2017 onwards when the number of studies increases, suggesting that the field is generally beginning to mature. For this reason, we analyze the evolution of the field based on the two time periods 2006-2016 and 2017-2020.

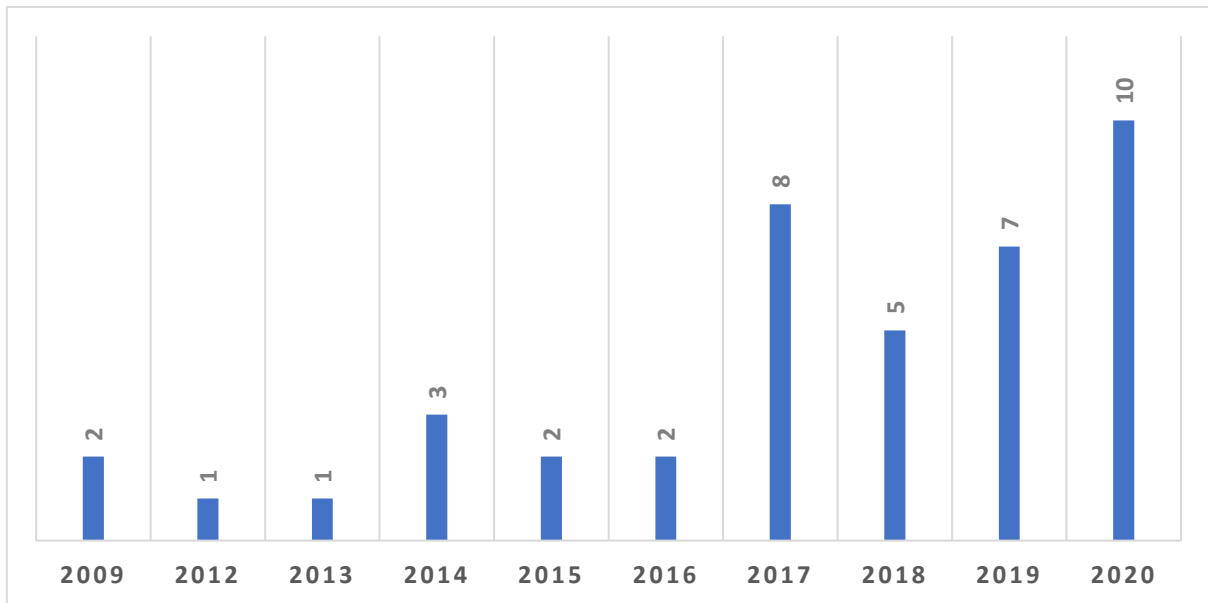


Figure 5-2. Studies by year

5.3.2 Interventions for CT development in higher education.

CT through programming empirical interventions in higher education (Table 5-2) mainly focus on Education majors, Natural Sciences majors and Computer Science (CS) majors. Table 5-3 presents the classification of branches based on the selected studies. The intense interest in Education branch led us to classify it as a separate branch in the context of this study. Figure 5-3 presents the percentage of studies by branch in periods 2006-2016 and 2017-2020.

Table 5-2. Interventions for CT development in higher education

Study	Content	Branch	Participants
(Adler & Kim, 2018)	Science methods course	Education	19 graduate and 13 undergraduate preservice teachers
(Bui et al., 2018)	Mindmaps and Scratch programming	Mathematics Education	50 preservice teachers
(Cachero et al., 2020)	Programming training	Health Information Systems, Psychology	104 undergraduate students
(Chao, 2016)	Principles and methods of C++ language programming	Information Communication	158 undergraduate students

(Choi, 2019)	Java programming class	Undefined	28 undergraduate students
(Cutumisu & Guo, 2019)	Educational Technology course	Education	139 preservice teachers
(Cetin, 2016)	Programming language course	Education	56 pre-service teachers
(Dolgopolovas & Jevsikova, 2015)	Structured programming (SP) course	Software Engineering	65 undergraduate students
(Fang et al., 2017)	Database Principles course	Computer Science and Technology	24 undergraduate students
(Fernandez et al., 2018)	Workshop	Education	21 in-service and pre-service teachers
(Fernandez et al., 2018)	Start to Programming course	Physics, Mathematics and Natural Sciences	22 undergraduate students
(Gabriele et al., 2019)	Programming course	Primary Education	141 preservice teachers
(Hambruch et al., 2009)	Introduction to CT	Physics and Chemistry	13 undergraduate students
(Hou et al., 2020)	Programming course	Beauty Science	40 sophomore students
(Jaipal-Jamani & Angeli, 2017)	Science education methods course	Elementary Teacher Education	21 preservice teachers
(Jeon & Kim, 2017)	CT-based programming course applicable to liberal arts	Education	110 preservice teachers
(Kang & Lee, 2020)	Project-based learning course	Non-engineering majors	Undergraduate students
(Kazimoglu et al., 2012)	Introductory computer programming	Computer Science	25 undergraduate students
(Katai, 2020)	Sorting algorithms	Humanities, Science	48 undergraduate students
(Kwon & Kim, 2018)	CT and Software Coding & Problem Solving and Algorithm courses	Humanities, Social sciences and Arts	250 undergraduate students
(Lee & Cho, 2020)	Computer programming	Undefined	151 undergraduate students
(Lin & Chen, 2020)	Program Logic Thinking Education	Arts, Music, Chinese, Public Administration	97 undergraduate students

(Mouza et al., 2017)	Integrating Technology in Education program	Education	21 preservice teachers
(Page & Gamboa, 2013)	How Computers Work: Logic in Action	Science, Engineering, History, Letters, Philosophy, Linguistics, Economics, Drama, Business, Psychology, Business, Computer Science, Computer Engineering	36 undergraduate students
(Pala & Mihçı Türker, 2019)	Programming-I	Education	33 preservice teachers
(Qin, 2009)	Introduction to Bioinformatics	Biology	Undefined
(Rodríguez-García et al., 2020)	AI, ML and its societal implications workshop	Computer Science	14 students
(Romero et al., 2017)	StorytoCode creative challenge	Elementary School Education	120 preservice teachers
(Rubinstein & Chor, 2014)	Computational Approaches for Life Scientists	Biology	25 graduate and undergraduate students
(Shih et al., 2015)	Computer Applications in Emergency Management	Emergency Management Technology	18 undergraduate students
(Wu et al., 2019)	Introduction to C++ programming	Education	47 preservice teachers
(Yuen & Robbins, 2014)	Introductory computer science course (data-driven)	Biology	5 undergraduate students
(Zha et al., 2020a)	Educational Technology course	Education	59 preservice teachers
(Zha et al., 2020b)	Educational Technology course	Education	15 preservice teachers

Table 5-3. Classification of branches

Branch sub-category	Description
----------------------------	--------------------

Majors (CS)	Computer Science including Computer Science and Technology, Computer Science, Computer Engineering, Software Engineering
Education majors	Education including Mathematics Education, Primary Education, Elementary School Education, Secondary Education
Non-majors in CS	Natural Sciences including Chemistry, Biology, Physics Humanities, Social sciences and Arts including History, Letters, Philosophy, Linguistics, Economics, Drama, Business, Psychology, Business, Arts, Music, Chinese, Public Administration. Engineering Mathematics Health Information Systems Information Communication Beauty Science

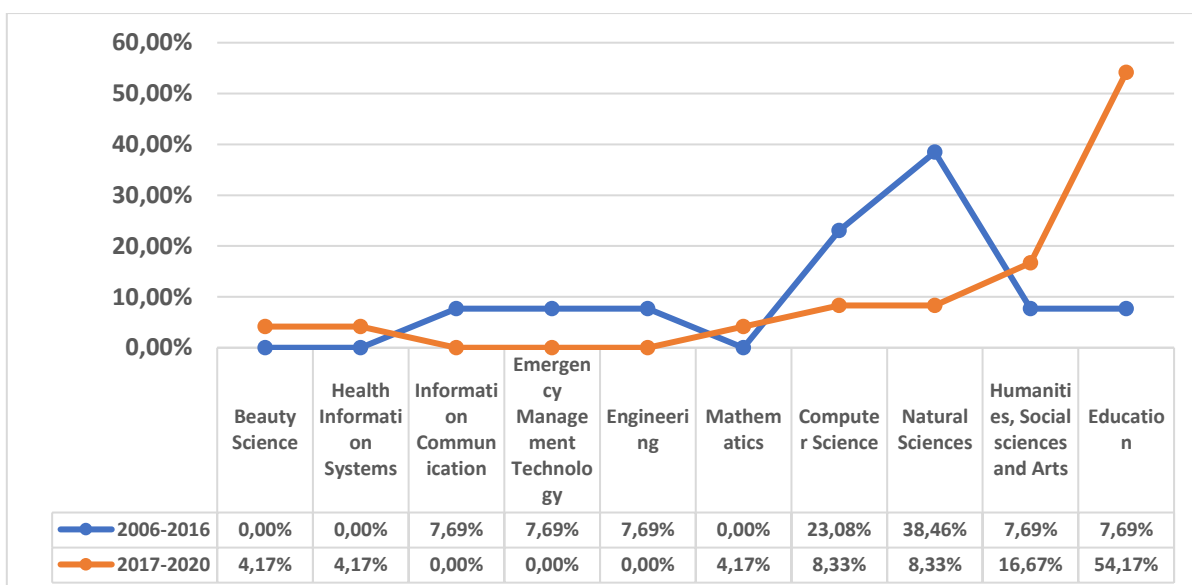


Figure 5-3. Percentage of studies by branch in periods 2006-2016 and 2017-2020

During period 2006-2016 a percentage of 69,23% focuses on Computer Science, Engineering and Natural Sciences while only 7.69% focuses on Education majors. During the next period 2017-2020 the focus shifts from the aforementioned branches to Education. A 54,17% of interventions for CT through programming focus mainly on preservice

teachers' preparation. Thus, we can conclude that there is an upward research trend for interventions aimed at Teacher Education.

5.4 The revised conceptual model for CT through programming in higher education (CTPHE)

The analysis of the selected studies revealed that the areas of Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building proposed by the CTPK-12 model also cover teaching and learning CT through programming in higher education. However, different sub-areas emerge in CT areas, while some of the model's sub-areas do not exist in the selected higher education studies. The following sections present the areas and sub-areas found in studies aimed at higher education. Figure 5-4 presents the revised CTPK-12 model that corresponds to CT through programming in higher education. The CTPK-12 model also depicts the relationships between the areas of teaching and learning CT through programming as links between the areas shown in Figure 5-4. The revised model could be use in order to develop research questions between the areas of teaching and learning CT through programming in higher education. For example, which learning strategies could be more appropriate for teaching CT domain specific elements, which for CT programming elements and which for CT higher-order skills.

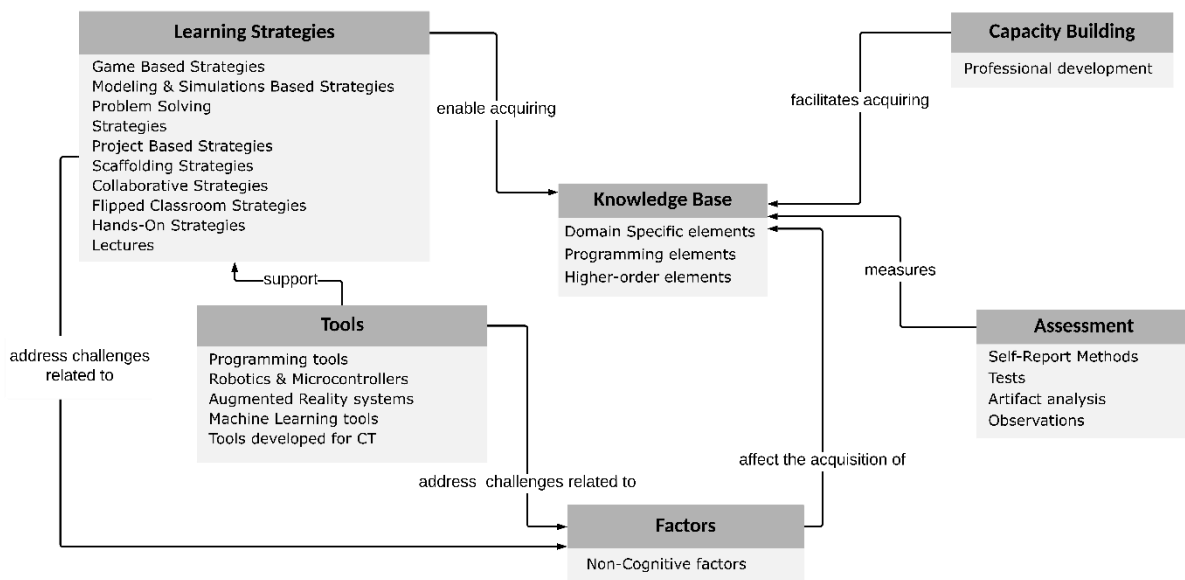


Figure 5-4. The revised conceptual model for CT through programming in higher education (CTPHE)

5.4.1 CT areas in higher education

The classification scheme identification phase revealed that there are no additional areas in the selected studies other than those indicated by the CTPK-12 model. Therefore, the areas of CT through programming in higher education which are analyzed and synthesized in the following sections are the following: Knowledge Base, Learning Strategies, Tools, Assessment, Factors, Capacity Building.

5.4.1.1 Knowledge Base

15 studies discuss elements of CT including domain specific elements, programming elements and higher-order skills. Table 5-4 presents the classification of CT elements in the selected studies. Figure 5-5 presents the distribution of CT Knowledge Base sub-categories by periods 2006-2016 and 2017-2020. Table 5-5 presents the distribution of CT Knowledge Base sub-categories by classified branch.

Chao (2016) investigates Computational practice (Sequence, Selection, Simple iteration, Nested iteration, Testing), Computational design (Problem decomposition, Abutment composition, Nesting composition) and Computational problem-solving performance (Goal attainment, Program size). Wu et al. (2019) adapts Brennan & Resnick's framework (2012), proposing Concepts (Sequence, Loops, Conditions, Operators, Data), Practices (Incremental and Iterative, Testing and Debugging, Reusing and Remixing, Abstracting and Modularizing) and Identities (Expressing, Questioning). In the same line, Cutumisu & Guo (2019) adopts Brennan & Resnick's framework (2012) for assessing CT concepts, practices and perspectives. Cetin (2016) investigates variables, conditional and selection statements, loops, arrays, and functions as CT elements. Yuen & Robbins (2014) investigates students' CT based on a coding scheme that includes Organization (Coding style, Data organization), Construction (Following procedures, Visualizing data) and Analysis (Interpretation and Conclusions). Jaipal-Jamani & Angeli (2017) investigate correct sequence, decisions on the flow of control and debugging.

Qin (2009) propose Multilevel abstraction and conceptualization, Iteration, recursion and backtracking, Modularization, Assessment and error corrections, Optimization and Simulation among other CT skill sets that are domain specific, derived from mapping CT skills to specific bioinformatics topics. In the same line, Rubinstein &

Chor (2014) propose Abstraction, Generalization, Modular design and decomposition, Data structures and Computational models among other domain specific computational concepts and processes.

Other studies propose skills such as Abstraction, Decomposition, Recognition of Patterns and Algorithms (Fernández et al., 2018; Hou et al., 2020), Creativity, Algorithmic Thinking, Cooperativity, Critical Thinking and Problem Solving (Korkmaz et al., 2017; Lin & Chen, 2020; Pala & Mıhçı Türker, 2019). Sondakh et al. (2020) propose a holistic CT framework that includes the skills of Abstraction, Algorithmic Thinking, Decomposition, Debugging, Evaluation, Generalization and the attitudes of Problem solving, Teamwork and communication.

Table 5-4. CT Knowledge Base sub-categories

Knowledge	Description	Studies
Base sub category		
Domain Specific elements	CT concepts, skills and processes mapped to specific domains.	PS31, PS34
Programming elements	Programming related concepts, practices, identities and designs.	PS4, PS5, PS7, PS11, PS15, PS38, PS39
Higher-order elements	Higher-order thinking skills and competencies.	PS10, PS13, PS21, PS24, PS29, PS36

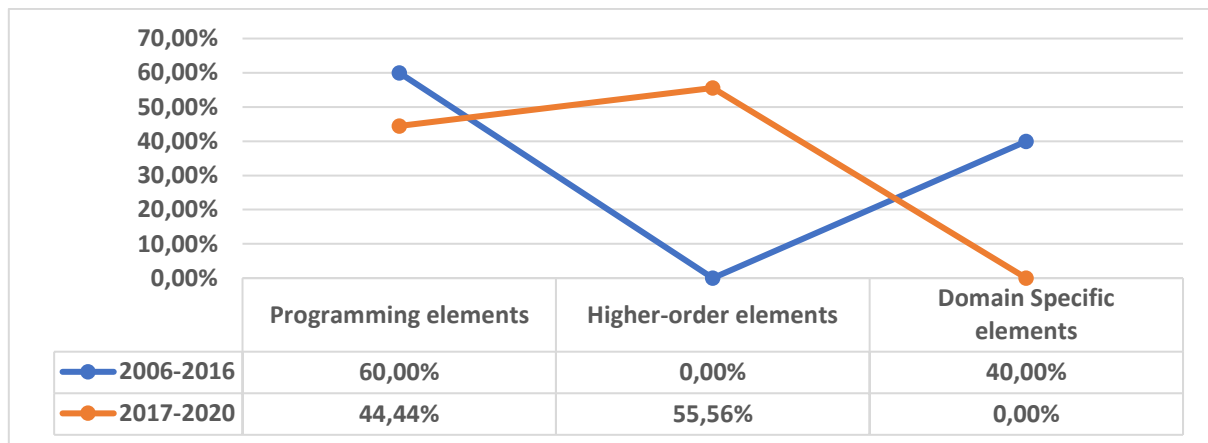


Figure 5-5. Distribution of CT Knowledge Base elements sub-categories by time period

Domain-specific elements are discussed in studies during period 2006-2016 while in period 2017-2020 these elements are absent. Higher-order elements are introduced during period 2017-2020 with a percentage of 60% in the selected studies of this period. Programming elements are discussed throughout the years.

Table 5-5. Percentage of studies' CT Knowledge Base elements sub-categories by classified branch

Knowledge Base sub-category	Non-majors in CS	Education majors
Programming elements	33,33%	66,67%
Higher-order elements	33,33%	33,33%
Domain Specific elements	33,33%	0,00%
	100,00%	100,00%

Domain Specific elements are discussed only in studies targeted non-majors in CS. Programming elements have the strongest presence in the selected studies and particularly in Education majors.

5.4.1.2 Learning Strategies

Researchers in 24 studies discuss, propose or apply teaching and learning strategies for CT through programming in higher education. Out of these studies, seven apply more than one learning strategy or practice. We classify learning strategies in nine sub-categories, namely, Game Based Strategies, Modeling & Simulations Based Strategies, Problem Solving Strategies, Project Based Strategies, Scaffolding Practices, Collaborative Strategies, Flipped Classroom, Hands-on strategies and Lectures. Table 5-6 presents studies by each sub-category. Figure 5-6 presents the distribution of learning strategies sub-categories by time periods 2016-2016 and 2017-2020. Table 5-7 presents the distribution of learning strategies sub-categories by classified branch.

Six studies discuss Problem Solving Strategies (Cetin, 2016; Hambrusch et al., 2009; Jeon & Kim, 2017; Kang and Lee, 2020; Lee & Cho, 2010; Yuen & Robbins, 2014). For example, Yuen & Robbins (2014) examine how undergraduate students develop CT skills during a data-driven programming course that encompasses problem-solving iterative processes. Lee & Cho (2020) exploit problem-solving methods to improve students' CT skills and logical thinking ability. Hambrusch et al. (2009) developed a course aimed at introducing students to CT based on a problem-driven format.

Four studies discuss Collaborative Strategies: Pair programming (Choi, 2019), Think-Pair-Share practice (Choi, 2019), Collaborative programming (Wu et al., 2019), teamwork (Jaipal-Jamani & Angeli, 2017; Zha et al., 2020b). Collaborative programming is proposed as an effective learning strategy to enhance students' CT in higher education (Wu et al., 2019). For example, Choi (2019) develops an instructional model that exploits Think-Pair-Share Strategy and pair programming. The results of this study show that collaborative strategies could help students learn CT and programming.

Three studies discuss Project Based Strategies (Ma et al., 2017; Wu et al., 2019). Wu et al. (2019) support that project-based learning contexts can help novice students develop different learning pathways to learn CT. In the same line, Ma et al. (2017) propose using project-driven learning strategies to enable students to acquire CT.

Three studies discuss Scaffolding strategies (Chao, 2016; Jaipal-Jamani & Angeli, 2017; Yuen & Robbins, 2014) usually combined with other strategies. Yuen & Robbins

(2014) propose scaffolding as an effective learning strategy in order to enable students to focus on higher-order computational concepts without struggling with coding process in a text programming language such as MATLAB. In the same line, Chao (2016) argues that scaffolding may facilitate students to develop programming strategies and skills. Jaipal-Jamani & Angeli (2017) also found that the scaffolding programming instructional strategy they applied in their study, helped students to acquire CT.

Two studies discuss Modeling & Simulations Based Strategies (Adler & Kim, 2018; Magana & Silva Coutinho, 2017), two studies Flipped classroom (Zha et al., 2020a, Zha et al. 2020b) and one study Game Based Strategies (Kazimoglu et al., 2012). Specifically, Kazimoglu et al. (2012) propose a serious game where students develop their game strategies through programming based on an educational game framework for CT.

Two researchers choose to give hands-on activities (Qin, 2009; Rubinstein & Chor, 2014) and three use lectures (Cetin, 2016; Gabriele et al., 2019; Jaipal-Jamani & Angeli, 2017). Other strategies involve reflective learning (Choi, 2019), storytelling (Romero et al., 2017) and network autonomous learning (Li & Hou, 2014). Additionally, learning strategies are implemented in traditional classroom settings or in blended environments (Fernández et al., 2018; Mouza et al., 2017; Zha et al., 2020b).

Table 5-6. Learning strategies sub-categories

Learning Strategies sub-category	Description	Studies
Game Based Strategies	Game Based Related Strategies involve game design and digital/video game development, programming games and any strategy that exploits games and programming.	PS18
Modeling & Simulations Based Strategies	Modeling & Simulations Based Related Strategies involve designing of scientific models and simulations.	PS1, PS26

Problem Solving Strategies	Problem Solving Related Strategies involve Problem Based Learning and problem-solving learning strategies in general.	PS4, PS12, PS16, PS23, PS26, PS39
Project Based Strategies	Project Based Related Strategies involve the engagement with authentic projects set around real challenges and problems.	PS26, PS38
Scaffolding Strategies	Scaffolding Related Strategies involve practices that offer support to students as they learn.	PS6, PS15, PS39
Collaborative Strategies	Collaborative Related Practices involve practices where students actively interact during the learning process including Pair programming, Think-Pair-Share practice and any practice based on student's collaboration and cooperation.	PS5, PS15, PS38, PS40
Flipped Classroom Strategies	Flipped classroom Strategies involve strategies that reverse the traditional model of classroom instruction.	PS40, PS41
Hands-On Strategies	Hands-on activities	PS31, PS34
Lectures	Theoretical lectures	PS4, PS11, PS15

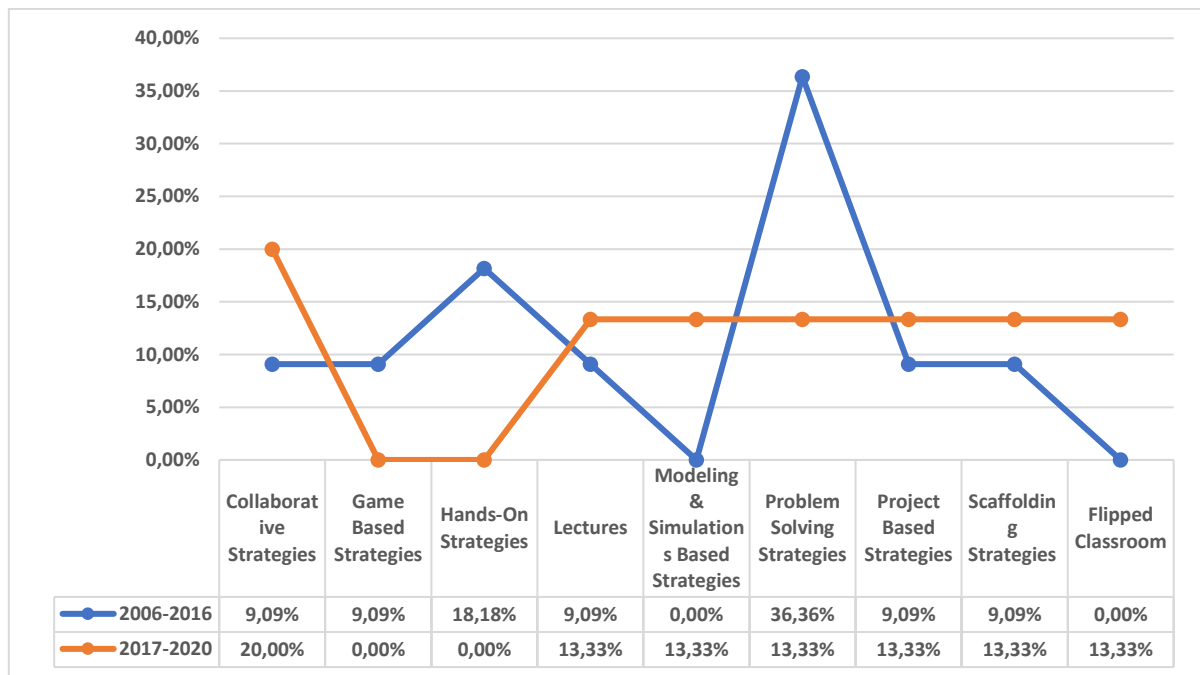


Figure 5-6. Distribution of learning strategies sub-categories by time period

During period 2006-2016 problem solving Strategies have the strongest presence (36.36%), while during period 2017-2020 almost all learning strategies sub-categories occupy the same percentage (13.33%) with the exception of Game Based Strategies which has no presence at all and Collaborative Strategies which have a slightly stronger presence than the rest (20%).

Table 5-7. Percentage of learning strategies sub-categories by classified branch

Learning strategies sub-category	CS majors	Educatio n majors	Non-majors
Collaborative Related Strategies	0,00%	23,08%	12,50%
Game Based Related Strategies	33,33%	0,00%	0,00%
Hands-On Strategies	0,00%	0,00%	25,00%
Lectures	0,00%	23,08%	12,50%

Modeling & Simulations Based Related Strategies	0,00%	7,69%	0,00%
Problem Solving Related Strategies	33,33%	15,38%	25,00%
Project Based Related Strategies	33,33%	7,69%	12,50%
Scaffolding Related Strategies	0,00%	7,69%	12,50%
Flipped classroom	0,00%	15,38%	0,00%
	100,00%	100,00%	100,00%

No strategy seems to be dominant in any of the classified branches. In addition, as shown in Table 5-7. Percentage of learning strategies sub-categories by classified branch, in studies aimed at preservice teachers and non-majors, a greater variety of studies is applied than in studies aimed CS majors.

5.4.1.3 Tools

Researchers in 37 studies discuss, propose or exploit tools for CT teaching and learning in higher education. We classify tools in five sub-categories, namely, Programming tools, Robotics & Microcontrollers, Augmented Reality Systems, Machine Learning tools and tools specifically developed for CT. Table 5-8 presents tools sub-categories leveraged in the selected studies. Figure 5-7 presents the distribution of tools sub-categories in periods 2006-2016 and 2017-2020. Table 5-9 presents the distribution of tools sub-categories by classified branch.

Eight studies exploit Scratch (Adler & Kim, 2018; Bui et al., 2018; Cetin, 2016; Gabriele et al., 2019; Hou et al., 2020, Mouza et al., 2017; Romero et al., 2017, Zha et al., 2020a), two studies Hour of Code (Adler & Kim, 2018; Mouza et al., 2017), one study Code.org (Cutumisu & Guo, 2019), one study App Inventor (Shih et al., 2015), one study

ARDUINO IDE (Pala & Mihçi Türker, 2019), one study LEGO® WeDo robotics (Jaipal-Jamani & Angeli, 2017), one study Java (Choi, 2019), one study Hopscotch (Zha et al., 2020b), one study HTML5 and CSS3 (Jeon & Kim, 2017) nine studies Python (Cachero et al., 2020; Dolgopolovas & Jevsikova, 2015; Hambrusch et al., 2009; Kang & Lee, 2020; Kwon & Kim, 2018; Lee & Cho, 2020; Magana & Silva Coutinho, 2017; Pala & Mihçi Türker, 2019; Rubinstein & Chor, 2014), one study ACL programming language (Page & Gamboa, 2013), two studies C++ (Pala & Mihçi Türker, 2019; Wu et al., 2019), three studies SQL (Huang & Leng, 2019; Qin, 2009; Fang et al., 2017), two studies MATLAB (Magana & Silva Coutinho, 2017; Yuen & Robbins, 2014), and four (Chao, 2016; Katai, 2020; Kazimoglu et al., 2012; Lin & Chen, 2020) studies develop a tool. For example, Chao (2016) develops a problem-solving programming environment and Lin & Chen (2020) develop a deep learning recommendation based augmented reality system.

Table 5-8. Tools sub-categories

Tools sub-category		Studies
Programming tools	Visual programming &	PS1, PS2, PS4, PS5, PS7, PS11, PS13, PS18, PS19, PS28, PS33, PS35, PS39, PS40
	Text programming tools.	PS3, PS6, PS8, PS9, PS12, PS14, PS16, PS17, PS22, PS23, PS27, PS29, PS30, PS31, PS34, PS35, PS38, PS39
Robotics & Microcontrollers		PS15, PS30
Augmented Reality systems		PS25
Machine Learning tools		PS32
Tools specifically developed to support a CT strategy		PS5, PS18, PS25, PS19

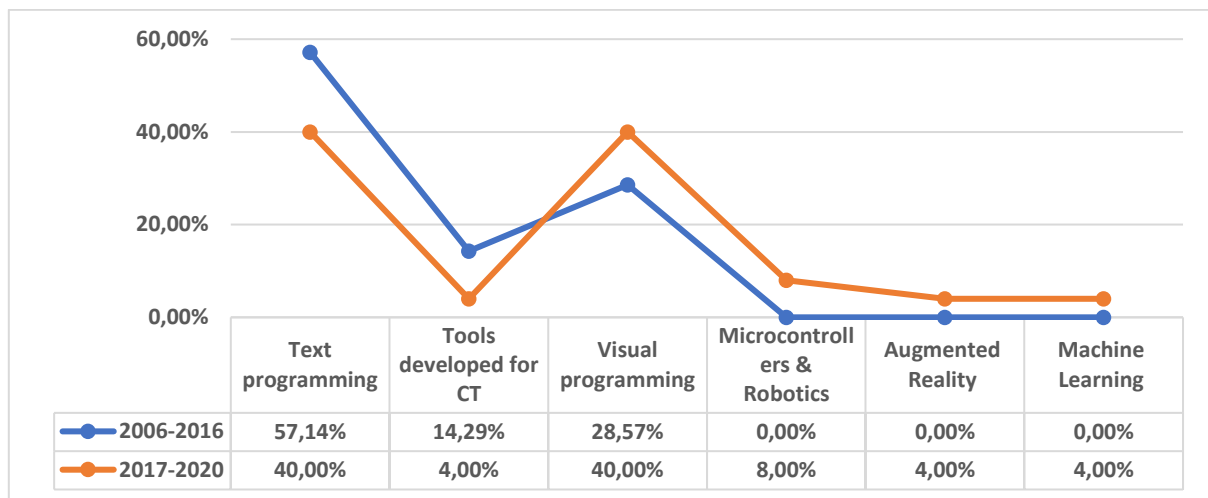


Figure 5-7. Distribution of tools sub-categories by period

During period 2006-2016 text programming tools have the strongest presence (57.14%) while 28.57% of studies investigates visual programming. Subsequently during period 2017-2020 a 40% of studies investigating visual programming. Thus, an upward trend in visual programming is revealed. In addition, new tools such as Microcontrollers, Robotics, Machine Learning tools and Augmented Reality systems are introduced.

Table 5-9. Percentage of tools sub-categories by classified branch

Tools sub-category	CS majors	Education majors	Non-majors in CS
Tools developed for CT	20,00%	0,00%	12,50%
Microcontrollers & Robotics	0,00%	13,33%	0,00%
Visual programming	20,00%	66,67%	25,00%
Text programming	40,00%	20,00%	56,25%
Augmented Reality	0,00%	0,00%	6,25%

Machine learning	20,00%	0,00%	0,00%
Total	100,00	100,00%	100,00%
	%		

Visual programming is investigated mainly in studies that focus on preservice-teachers while it is not prevalent in studies that target Non-majors and CS majors. Text-programing is investigated in all branches while it is prevalent in studies that target Non-majors in CS (56,25%) and CS majors (40%).

5.4.1.4 Assessment

29 studies discuss CT through programming assessment methods. Assessment methods are classified in four sub-categories, namely, Self-report methods, Tests, Artifact analysis and Observations. Table 5-10 presents assessment methods applied in the selected studies. Figure 5-8 presents the distribution of assessment sub-categories in periods 2006-2016 and 2017-2020. Table 5-11 presents the distribution of assessment sub-categories by classified branch.

Four of the selected studies involve observations. Wu et al. (2019) record students' actions and conversations (screen and video recording) to examine how novice programmers develop CT by interacting with each other during collaborative programming and problem solving. More specifically, they investigate students' trajectories and their different CT development pathways. Screen recording is used to capture the programming process while video recording is used to capture student's conversations. Yuen & Robbins (2014) collect field notes during participants interviews.

Six of the selected studies involve artifact analysis. Chao (2016) collects log data about the participants' practice, strategies, and performance of computational problem-solving activities. Choi (2019) evaluates students' programming artifacts. Yuen & Robbins (2014) collect source code from students' in-class activities. Romero et al. (2017) analyze students' projects through Dr. Scratch (Moreno-Leon et al., 2015) and manual inspection based on entities, events, code blocks and errors. Gabriele et al. (2019) analyzed students' Scratch files through manual inspection for programming concepts, code organization and

designing for usability adapted by Denner et al. (2012) and automatic inspection through Dr. Scratch.

23 studies exploit self-report assessment methods. Five studies exploit scales, three surveys, seven interviews, eight questionnaires and one study students' reflections. Yuen & Robbins (2014) use interviews as their primary method for data collection. Shih et al. (2015) survey students' perceptions about programming and their experiences with the applied CT intervention. Mouza et al. (2017) assess students' CT knowledge based on a pre/post scale. Cutumisu & Guo (2019) used topic modeling techniques to extract participants CT understanding through their reflections. Researchers also develop and validate self-report scales in their studies. For example, Korkmaz et al. (2017) developed the CTS scale in order to assess students' CT skills. The scale includes the items of Creativity, Algorithmic Thinking, Critical Thinking, Problem Solving and Cooperativity. Sondakh et al. (2020) propose a scale for CT assessment validated through Fuzzy Delphi Method that includes the items of Abstraction, Algorithmic Thinking, Decomposition, Debugging, Evaluation, Generalization, Problem solving, Teamwork, Communication and spiritual intelligence. In the same line, Kılıç et al. (2020) developed and validated a scale that includes the items of Conceptual Knowledge, Algorithmic Thinking and Evaluation. Finally, ten studies assess students' CT through tests and assignments. For example, Jaipal-Jamani & Angeli (2017) used programming worksheets with completed, semi-completed and new programming tasks. Lin & Chen (2020) used multiple-choice and fill-in-the-blank questions to assess students' programming understanding.

Table 5-10. Assessment sub-categories

Assessment sub-category	Description	Studies
Self-Report Methods	scales, questionnaires, surveys, interviews, reports, reflections	PS1, PS2, PS3, PS4, PS5, PS7, PS10, PS11, PS15, PS16, PS17, PS20, PS21, PS22, PS28, PS29, S30, PS31, PS34, PS35, PS36, PS39, PS40

Tests	multiple choice and open-ended tests, quizzes, tasks, assignments	PS3, PS4, PS15, PS18, PS25, PS31, PS32, PS34, PS39, PS40
Artifact analysis	automatic analysis, manually inspection of artifacts, log data	PS5, PS11, PS19, PS33, PS38, PS39
Observations	observations of students' actions, screen recordings, camera recordings, field notes	PS2, PS37, PS39, PS40

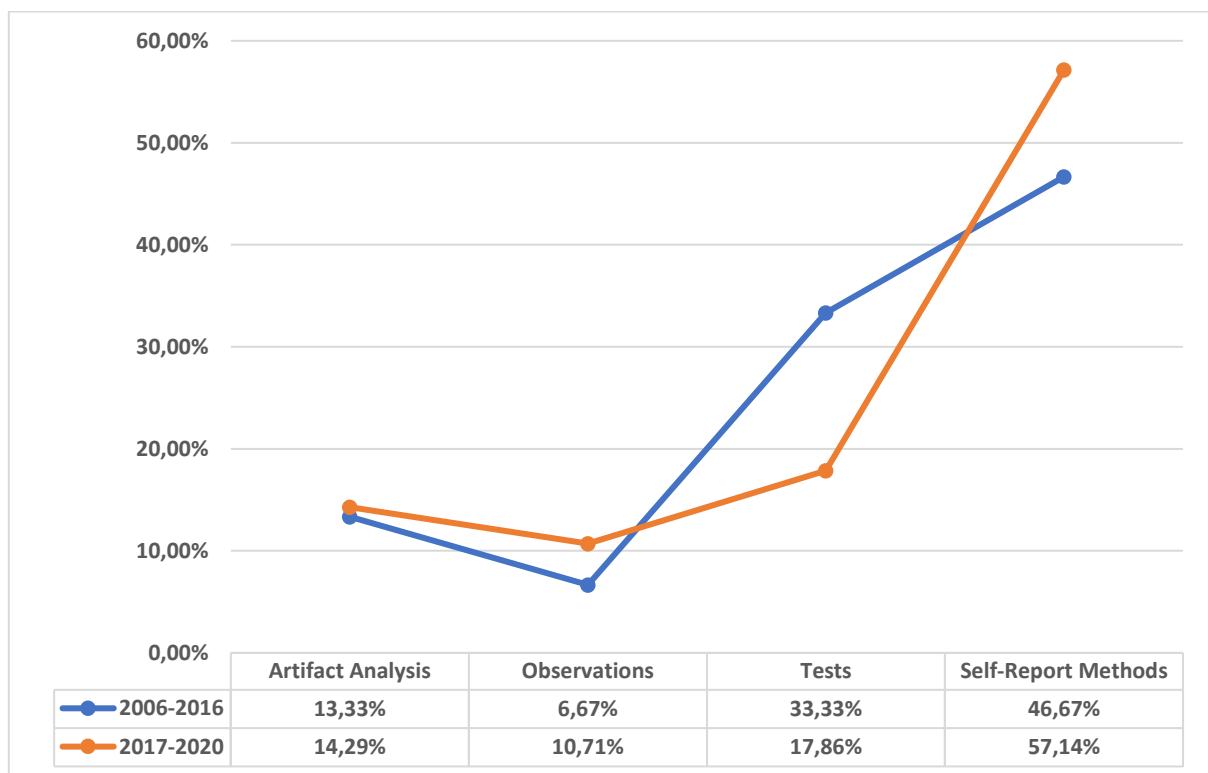


Figure 5-8. Distribution of assessment sub-categories by period

During period 2017-2020 an upward trend in the use of observations (+4,04%) and self-report methods (+10,47%) and a downward trend in the use of tests (-15,47%) is revealed in the assessment of CT. Artifact analysis shows a very small increase of 1,48%.

Table 5-11. Percentage of assessment sub-categories by classified branch

Assessment sub-category	CS majors	Education majors	Non-majors in CS
Artifact Analysis	0,00%	15,79%	14,29%
Observations	0,00%	15,79%	7,14%
Tests	50,00%	10,53%	28,57%
Self-Report	50,00%	57,89%	50,00%
Methods			
Total	100,00%	100,00%	100,00%

Self-report methods have the strongest presence compared to other methods in studies targeted Non-majors in CS (50%) and education majors (57.89%).

5.4.1.5 Factors

Nine studies discuss factors that affect CT. Table 5-12 presents factors discussed in the selected studies. The effects that CT could have on interest in Computing and attitudes toward programming (Cetin, 2016; Hambrusch et al., 2009; Shih et al., 2015), self-efficacy (Jaipal-Jamani & Angeli, 2017; Kwon & Kim, 2018), creativity (Romero et al., 2017), interest in CT (Zha et al., 2020a), motivational impact (Katai, 2020), enrollment in CS courses (Hambrusch et al., 2009) and occupational change (Kwon & Kim, 2018) are discussed in the selected studies. CT-related factors are discussed through the years, 33.33% of the studies are published during 2006-2016 and another 66.67% during 2017-2020. Furthermore, studies that investigate CT-related factors focus on both Education Majors (57.14%) and Non-majors in CS (71.43%).

Hambrusch's et al. (2009) study reveals that the problem-driven approach focused on computational principles and scientific discovery they applied, increased students' interest in CS. In the same line, Shih et al. (2015) found a positively change in students' perceptions about computing after they attended a course aimed to encourage students to

apply CT and problem-solving skills to authentic problems. On the contrary, Cetin (2016) found no significant difference between control and experimental group students in terms of their attitudes towards programming. However, he suggests that this is probably due to the short duration of the intervention and the difficulty of changing students' already high attitudes. Kwon & Kim (2018) conclude that a software education curriculum based on CT can stimulate students' intrinsic motivation and improve students' self-efficacy. In the same line, Jaipal-Jamani & Angeli (2017) found that after participated in a CT robotics program students' self-efficacy related to robotics and interest in learning robotics significantly increased. Kwon & Kim's (2018) study reveals that integrating CT could affect students' perspectives about their future occupation.

Table 5-12. Factors investigated in the selected studies

Factors	Description	Studies
Non-Cognitive factors	Personal traits, attitudes and motivations such as attitudes toward programming, self-efficacy, creativity, interest in CS, perspective about future occupation.	PS4, PS12, PS15, PS19, PS22, PS27, PS33, PS35, PS41

5.4.1.6 Capacity Building

Only three of the selected studies discuss academic faculty training and professional development and they are all published in period 2017-2019. Table 5-13 presents methods regarding capacity building discussed in the selected studies.

Magana & Silva Coutinho (2017) survey industry and academia experts to identify the challenges facing academic staff in integrating CT at undergraduate level. Ma et al. (2017) suggest ways to improve university student's CT skills, including faculty professional training based on two principles: the mobility of academic staff and the organization of training programs. Taylor et al (2018) emphasize the role of collaboration between institutions as a means of motivating academic staff to redesign courses to integrate new concepts such as CT and coding.

Table 5-13. Capacity Building methods

Capacity Building	Description	Studies
Professional development	Variety of tools such as training programs, mobility of academic staff, collaboration between institutions.	PS26, PS27, PS37

5.5 Discussion

The analysis of the selected studies revealed that the areas of Knowledge Base, Learning Strategies, Assessment, Tools, Factors and Capacity Building proposed by the CTPK-12 model also cover teaching and learning CT through programming in higher education. However, different sub-areas emerge in CT areas, while some of the model’s sub-areas do not exist in the selected higher education studies.

Furthermore, as CT applications become more mature these areas evolve. Early attempts often link CT to domain-related elements, drawing on topics and activities related to specific courses and disciplines. However, in the coming years, CT is considered as a construct that is more associated with high-level skills such as abstraction and decomposition. Elements related to programming are most prevalent and evident throughout the years. This is plausible as CT draws from CS concepts according to Wing’s (2006) definition.

CT through programming in higher education is traditionally implemented through text programming environments. However, the analysis of the selected studies revealed an upward trend in visual programming. This could be explained as visual programming is often applied to teacher education courses that have been at the forefront of CT higher education in recent years. In addition, tools such as Microcontrollers, Robotics and Augmented reality systems have recently emerged.

CT assessment is generally considered difficult to achieve by several authors (Brennan & Resnick, 2012; Denning, 2017; Fronza et al., 2017; Werner et al., 2012; Zhong et al. 2016). While self-report methods are the most common, the analysis of the selected

studies also revealed a shift from tests to artifact analysis and observations in recent years. These methods are incorporated in order to provide a more complete picture of the CT acquisition.

Learning strategies and factors related to CT development such as personal traits, attitudes and motivations are discussed throughout the years, while academic faculty training and professional development gained attention only recently.

Teaching and learning CT through programming in higher education could be also organized in two areas: CT development for Non-majors and CS majors; and Teacher Education. The first concerns interventions and studies that propose the integration of programming aiming to help Non-majors and CS majors to acquire CT. The second concerns efforts to educate and support pre-service teachers with ultimate goal the integration of CT in K-12 education. The two areas present differentiation mainly in the tools used and the CT elements that are assessed with the second one to draw upon research on CT conducted in K-12 settings. Implementation of CT through programming for pre-service teachers is designed mainly on the basis of programming elements and includes mainly visual programming.

The analysis of the selected studies reveals that the focus of CT research in higher education is mainly on re-designing courses to align disciplinary knowledge with CT core concepts and to provide instructional models. The development of frameworks for learning strategies, tools and assessment methods is not extensively discussed in the selected studies.

Herein we also identify gaps that we discuss in the following paragraphs in an attempt to draw connections and implications from K-12 education where extensive efforts are being made worldwide to integrate CT.

In terms of learning strategies, although previous research has revealed that game design is often selected to introduce software engineering to students Souza et al. (2018), this is not the case for CT in higher education. There is no study in the selected studies that focuses on the development of CT through programming that applies game design learning strategy. In contrary, in K-12 education, game design is one of the most common strategies applied in several studies such as (Garneli & Chorianopoulos, 2018; Repenning et al.,

2015; Weintrop et al., 2016). This is probably due to the capabilities of the tools offered to different age groups. In K-12 education, various tools such as Scratch (Resnick et al., 2009), and Agentsheets (Repenning et al., 2015) are utilized for game design and media computation, supporting the implementation of learning strategies that include game design learning. Although these tools are widely used in K-12 education and in higher education to prepare future teachers (Adler & Kim, 2018; Angeli et al., 2016; Gabriele et al., 2019), they are rarely used in interventions targeted other CS major or non-major students. Text programming languages that are mainly used in higher education pose challenges to students such as dealing with complex syntaxes and abstract concepts (Buitrago Flórez et al., 2017) and require deep programming learning and experience to enable students to develop a game.

The importance of learning strategies in CT development is emphasized in both K-12 and higher education studies. Denner et al. (2012) study reveals that introducing CT to young students without applying a learning strategy, causes difficulties in developing students' CT skills. In the same line, Dolgopolas & Jevsikova, (2015) argue that appropriate learning strategies should be exploited in order to facilitate CT skills development. They suggest that programming didactical approaches in higher education should focus on problem solving skills rather than language programming syntax.

Only few studies (Lee & Cho, 2020; Li & Hou, 2014; Ma et al., 2017) focus on creating frameworks by aligning learning strategies with CT. The bulk of research in higher education focuses on the implementation of learning strategies within specific courses and the development of instructional models.

Although there are studies that underline the role of communities in CT development (Xing, 2019) and the need to shift from tools to Communities (Clark & Sengupta, 2019; Kafai, 2016), as CT and programming are social practices, the exploitation of programming Communities in higher education is still lacking behind. Content-specific tools and mainly text programming languages are those applied in the higher education context. This in line with Magana & Silva Coutinho's (2017) study, showing that tools for teaching and learning CT in higher education are chosen on the basis of subjects rather than on their ability to support the acquisition of these skills. Exception are studies that focus on pre-service teachers that investigate mainly visual programming.

CT assessment in higher education applies the same assessment methods (Artifact Analysis, Observations, Tests and Self-report) as in K-12 education. However, the assessment is mainly carried out in the context of course evaluation. There are some efforts to develop universally accepted assessment methods but all of them are self-report methods. This is consistent with Lyon and Magana (2020) review that highlights the strong presence of self-report assessment methods in higher education CT studies. In addition, studies do not always attempt to validate the methods used and often do not yield quantitative results. Other challenges involve the small sample size and the lack of CT specific elements in the studies' results.

Moreover, often while studies present in the background various definitions of CT, they do not ultimately provide information on which elements of CT they focus on based on these definitions. Many times, they do not mention the CT context on which they are based, or display CT elements that are not based on a clear definition, are poorly documented and often vague.

Females and minority groups are often underrepresented in computing, as well as in technology labor (Jenson & Droumeva, 2016). Cooper et al. (2014) suggest that research in computing education should focus on gender and other minority groups. In addition, Shute et al. (2017) review the literature highlighting that researchers consider utilizing CT to motivate learners, especially females and minorities. However, there are limited studies (e.g., Zha, 2020a) in higher education that discuss the use of CT through programming to address issues related to female or underrepresented students. In addition, although gender as a factor affecting CT acquisition is particularly discussed in K-12 education (Atmatzidou & Demetriadis, 2016; Durak & Saritepeci, 2018), this is not the case for higher education. Studies in higher education do not focus on examining the relationship between gender and other social factors with CT.

Although teachers' knowledge and needs and their preparation to support students' understanding of CT are highly discussed in K-12 literature (e.g., Alfayez & Lambert, 2019; Angeli et al., 2016; Bower et al., 2017; Giannakos et al., 2015; Israel et al., 2015; Mouza et al., 2017; Yadav et al., 2017), research in higher education rarely focuses on faculty preparation. Only two of the selected studies involve higher education faculty

(Magana & Silva Coutinho, 2017) or discuss opportunities for professional development (Ma et al., 2017).

5.6 Summary

This chapter presents the method and results of the second phase of this dissertation that involves (a) the study of the areas and relationships of the CTPK-12 conceptual model in the context of higher education and (b) the investigation of these areas based on the following two dimensions: i) their evolution over the years and ii) the branches to which CT is applied. For this purpose, a systematic mapping methodology was applied. Main results include the identification of the CT areas of Knowledge Base, Assessment, Learning Strategies, Tools, Factors and Capacity Building. Of these, Knowledge Base, Assessment and Tools have significantly evolved throughout the years, while Capacity Building has only recently emerged. In addition, the introduction of CT to undergraduate students and pre-service teachers differs mainly in the tools used and the CT elements that are assessed.

6 Designing and evaluating a Computational Thinking tool

6.1 Introduction

In this Chapter, we proceed to the design and evaluation of a Computational Thinking tool. The implementation of the tool was done by Maria Mousiou during her master thesis (Mousiou, 2021). We design a Computational Thinking game that incorporates scaffolding features and can further be parameterized to produce different versions that are used in the study presented in the next chapter. In addition, we evaluate the game and investigate the perceived effectiveness of its scaffolding features.

The remainder of this Chapter is organised as follows: Section 6.2 presents the study design. Section 6.3 presents the Scaffolding Computational Thinking tool. Section 6.4 presents the evaluation of the tool and the perceived effectiveness of its scaffolding features. Section 6.5 further discusses the chapters results. Section 6.6 presents a summary of the chapter.

6.2 Study design

6.2.1 *Study goal and research questions*

This study aims to design and evaluate a Scaffolding Computational Thinking game.

The research questions of the study are:

RQ1. Do students perceive the aMazeD Scaffolding Computational Thinking Game as ease to use?

RQ2. Do students perceive the aMazeD Scaffolding Computational Thinking Game as effective on learning Computational Thinking?

RQ3. Do students perceive the scaffolding features of the aMazeD Scaffolding Computational Thinking Game as effective in learning Computational Thinking?

6.2.2 Research design and Participants

To answer the research questions of the study, we adopted a survey research approach. For this purpose, we designed and conduct a study in a Greek school for students from grades 7 to 9 (ages 13 to 15) that has been approved by the Ethics Committee of the university of the authors. The study was conducted during formal teaching hours and lasted one and a half hours (two teaching hours) for each grade. Students played the Scaffolding Computational Thinking Game for one hour and subsequently were asked to complete a questionnaire for about 30 minutes. Only students whose parents gave their written consent participated in the intervention. A total of 28 students were finally participated in the study.

6.2.3 Instrument

We adapted the instrument (Appendix C) developed by Park (Park, 2009) which is based on the technology acceptance model (TAM), in order to use it for the data collection.

The questionnaire is divided in the following sections:

- Perceived ease of use (PE)
- Perceived usefulness (PU)
- Attitude (AT)
- Accessibility (AC)

A 5-point Likert scale from 1 to 5 was used for, where 1 equal “Strongly Disagree”, 2 equals “Disagree”, 3 equals “Indifferent”, 4 equals “Agree” and 5 equals “Strongly Agree”.

In addition to the above sections, a demographics section was included, as well as an open-ended question about the overall experience.

6.2.4 Study Limitations

We acknowledge that this study has some limitations. First, the study is designed to include only one research group. Second, the results are based on a self-report measure and capture student’s opinions and perceptions.

6.3 The aMazeD Scaffolding Computational Thinking Game

6.3.1 aMazeD General Description

The “aMazeD” scaffolding Computational Thinking game (Mousiou, 2021) consists of 10 levels, most of which are adaptations of the Computational Thinking Test (CTt) questions developed by Román-González et al. (2017). The game is developed with Blockly and is based on Blockly Games Maze and Turtle. Each level belongs to one of the following categories: a) Maze and b) Turtle.

On the one hand, the goal of the game in the levels belonging to the maze category is to guide the avatar from the beginning to the end following a certain path. On the other hand, the goal of the game at the levels belonging to the turtle category is to guide the avatar to draw the required shapes. In both cases, the player uses programming blocks to give the appropriate instructions to the avatar to complete the levels.

The game environment consists of the following parts: the navigation bar, the instruction bar, the main game frame, the results box, the Blockly toolbox and the workspace (Figure 6-1).

The level numbers are displayed in the navigation bar. The light purple color indicates the completed levels as well as the current level at which the user is. The player does not have the right to move to any level of his/her choice. The player starts from level 1 and with the submission of his/her answer moves to each next level. In the left part of the navigation bar the player can select the desired language. In the workplace the player can stack the blocks in order to create the program that will finally solve the level. The Blockly toolbox contains the available blocks for each level.

Level instructions:

Correct the block options if needed and put the right statements in the if-else block in order to help me reach my goal.

Figure 6-1. The aMazeD game environment

Below the main game frame there are the three buttons “Play”, “Reset” and “Submit”. By clicking the play button, the player can see the visual execution of the code inserted in the workspace. During code execution, the executed blocks are highlighted. No level output is displayed after the execution. The play button allows students to see the execution of their designed solutions, try them out and debug their code. By clicking the reset button, the character or brush moves to the beginning of the path or to the beginning of the shape. The game is restored to its original state. No code execution is taking place.

By clicking the submit button the player can see the movement of the character or the brush depending on the instructions loaded in the workspace. During code execution, the executed instructions are highlighted. After the execution, a message is displayed with the level output. If the player manages to solve the level, a success message is displayed, otherwise a failure message is displayed. In both cases, the submitted instruction is translated to JavaScript and displayed in the screen, while the game moves to the next

level. The player is transferred to the next level regardless of whether the current level has been completed successfully.

In the results box the message Success or Failure is displayed, in addition to the current level score and the player's overall score up to that level. The level output and the score of the level are displayed after the submission by the player. Furthermore, some additional information is displayed such as the time needed for completion and the times that the paly button was pressed.

6.3.2 Computational Thinking Concepts and Practices Covered by the Scaffolding Computational Thinking Game

The player must employ different Computational Thinking concepts and practices according to Brennan’s and Resnick’s framework (Brennan & Resnick, 2012) in order to solve each level. Computational Thinking concepts and Practices covered by the game are presented in Table 6-1.

Table 6-1. CT Concepts and practices per aMazeD level

Computational Thinking concepts adopted from Brennan and Resnick (2012)		
Concept	Description	Application to aMazeD levels
Sequences	Basic instructions and directions	The player needs to design a sequence of steps in order to solve the level (Level 1,7)
Loops	Repeat a set of instructions for a specific number of times or until a condition becomes true	The player needs to repeat a set of instructions in order to solve the level (Level 2-6, 8-10)
Conditionals	Constraints that allow the execution of different instructions	The player needs to design a solution that involves the selection of a choice based on constraints (4-6)

Computational Thinking practices adopted from Brennan and Resnick (2012)		
Practice	Description	Application to aMazeD levels
Testing and debugging	Trial and error processes for correcting malfunctions	The player needs to make corrections to a given set of instructions (Level 1-3, 5, 7)
Being incremental and iterative	Design and implement solutions using iterative processes	The player uses the play button in order to see the execution of the game and make changes to his/her solution until the final submission (Level 1-10)

6.3.3 aMazeD Scaffolding Features

The aMazeD game is designed and developed to support scaffolding based on a three-dimension framework that includes: i) the provision of a semi-finished or semi-correct solution, ii) instructions and explanations of the Computational Thinking concepts required for the solution of the level and iii) the provision of support regarding the logic behind the solution design.

The scaffolding game provides semi-finished preloaded workspace solutions for each level. This aims to make it easier for students to understand and use the concepts of Computational Thinking as they are asked to make small changes to pre-existing semi-finished solutions rather than writing their own from scratch (Werner et al., 2012). In addition, the player has the ability to run the semi-finished solutions before even trying to solve the level so as to observe exactly how the avatar moves with the given instructions. In this way, he/she can better and more deeply understand how Computational Thinking concepts such as sequence, loops and conditions work. When the solution is executed for the first time, an explanation of the Computational Thinking concepts covered at the level is displayed. The explanation concerns the operation and use of the specific concepts of Computational Thinking. Subsequently, when the solution is executed for the second time, a prompt about the logic behind the solution of the level appears. This way, the player

could understand how he could use Computational Thinking concepts to solve the particular level.

Following the above framework, we construct scaffolding for students, first ensuring the understanding of the concepts of Computational Thinking by providing them with incomplete solutions and explanations regarding the use of the concepts. We then provide support to students to help them understand how they could use these concepts to design effective solutions.

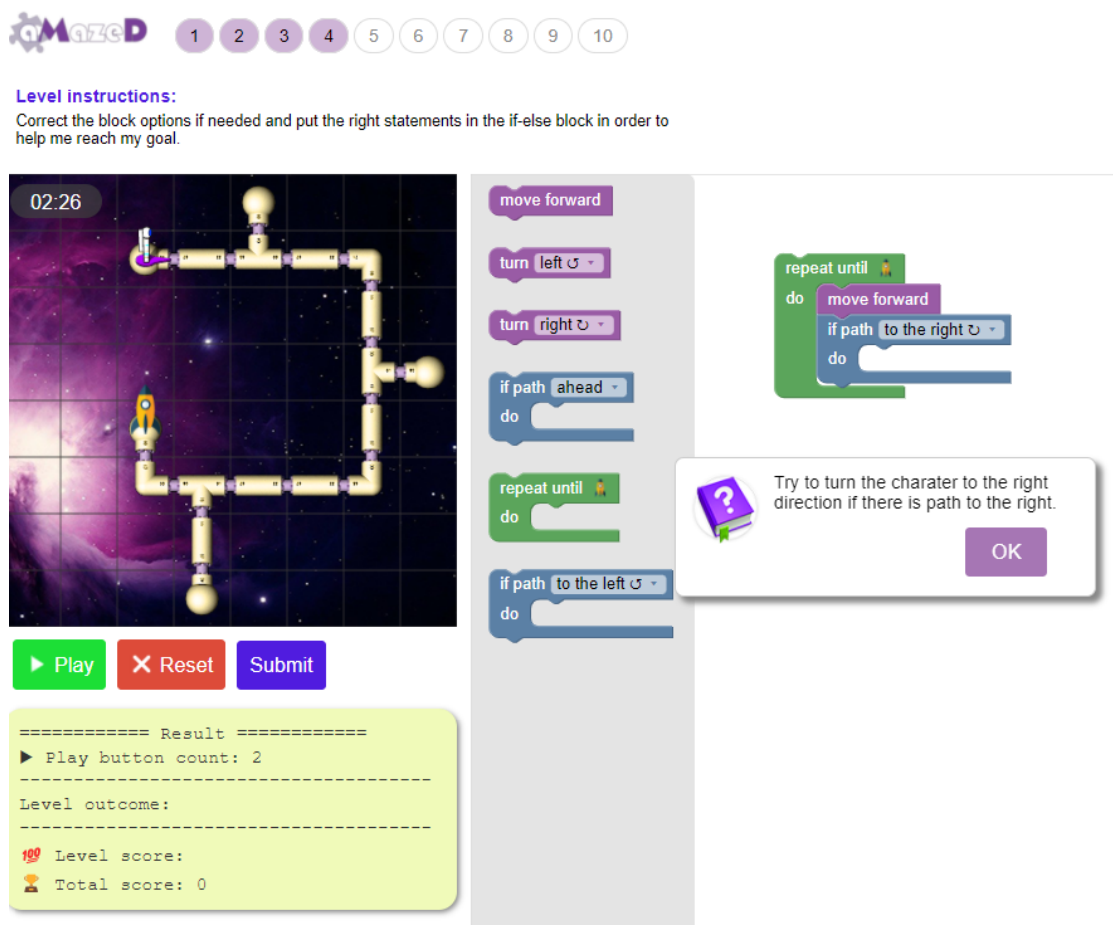


Figure 6-2. Semi-finished instructions

All game levels are designed based on the three-dimension framework described above. In the following paragraphs we present how the aforementioned framework is applied at Level 4.

A semi-finished solution appears in the workspace when the level is loaded (Figure 6-2). In addition, an instruction for correcting the given solution appears. When the

solution is executed for the first time, the following explanation about the “if” block is displayed: “The 'if' block will execute the 'do' block only if the condition is true. Subsequently, when the solution is executed for the second time, the following prompted is displayed: “Try to turn the avatar to the right direction if there is path to the right.”.

6.3.4 aMazeD Analytics Features

Logs are kept for assessment and self-assessment purposes. The data is stored locally at browser level and displayed on the results page where the teacher or student can download it in pdf or excel format. Except the total score of the game the following data is stored for each level:

- Level outcome: the result of the level, Success or Failure
- Score: the level score, zero if it was a failure
- Time: the time it took the player to click "Submit" button
- "Play" button: how many times the “Play” button was pressed
- JavaScript code: the code submitted.

6.4 Results

6.4.1 Demographic Data of the Participants

The students who participated in the study are in grade 7, 8 and 9. A high percentage of students (75%) stated that they have previous programming experience. This is important as they may be able to compare their previous experiences with the experience from the Scaffolding Computational Thinking Game and draw safer conclusions about it. The demographic data of the participants are presented in Figure 6-3.

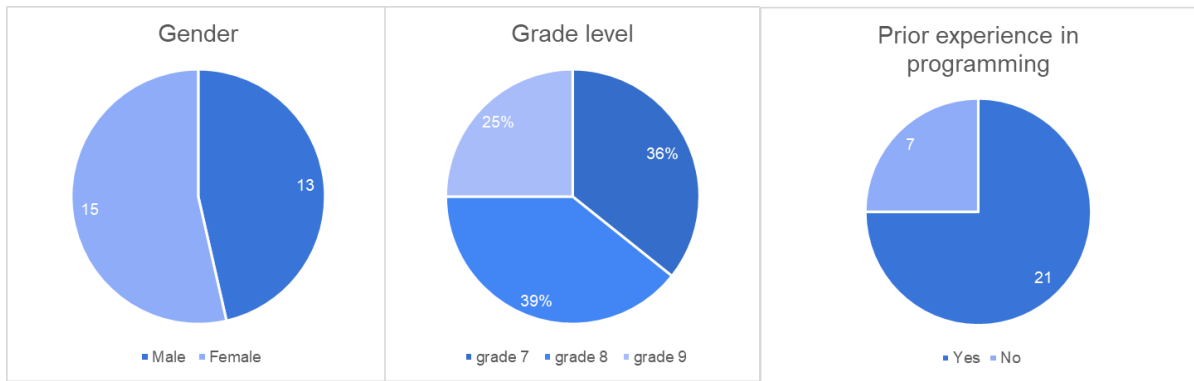


Figure 6-3. Demographic data of the participants

The scale had a good level of internal consistency, as determined by a Cronbach's alpha of 0.761. The following paragraphs present the results of each section of the scale.

6.4.2 Perceived ease of use (PE)

PE1. I find the aMazeD programming and Computational Thinking game easy to use.

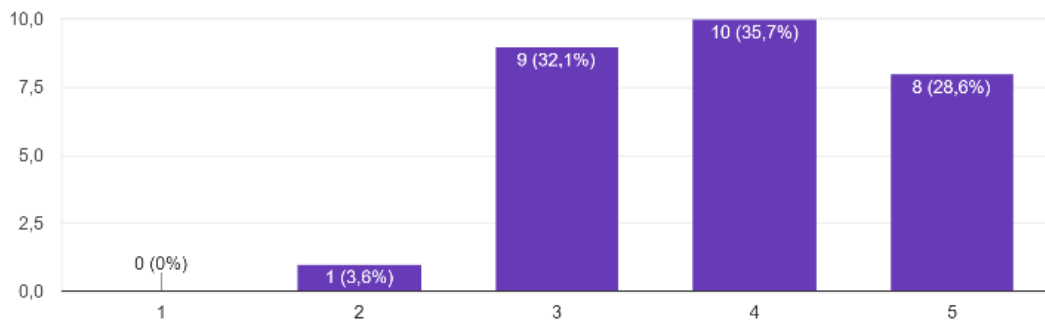


Figure 6-4. Results on PE1

PE2. Learning how to use a programming and Computational Thinking game is easy for me.

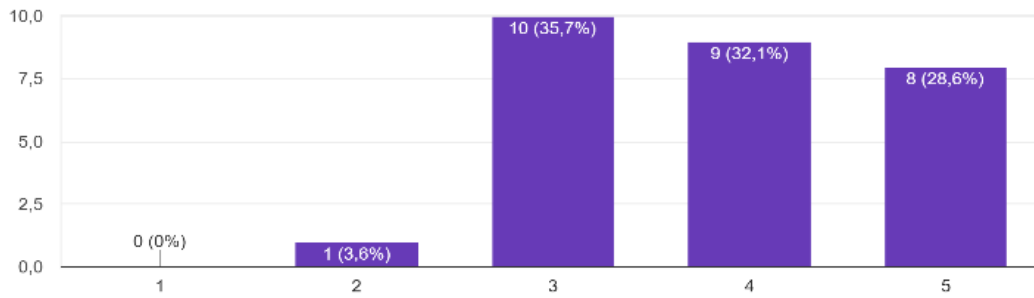


Figure 6-5. Results on PE2

Figure 6-4 and Figure 6-5 present the students' answers to PE1 and PE2 respectively. 64,3% of the students perceived the game as easy/very easy to use which is a slightly higher than the 60,7% who answered that they find easy/ very easy to learn how to use a Computational Thinking game. While only 3,6% answered that disagrees that the game is easy to use.

6.4.3 Perceived usefulness (PU)

PU1. The aMazeD game would improve my understanding of the concepts and practices of programming and Computational Thinking.

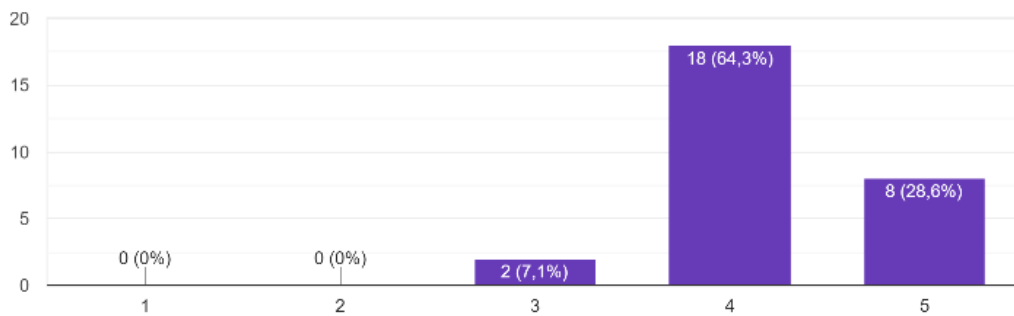


Figure 6-6. Results on PU1

Figure 6-6 presents the students' answers to PU1. A high percentage of 92.9% of the students answered that the aMazeD game would improve their understanding of Computational Thinking practices.

PU2. The aMazeD game could make it easier to study the concepts and practices of programming and Computational Thinking.

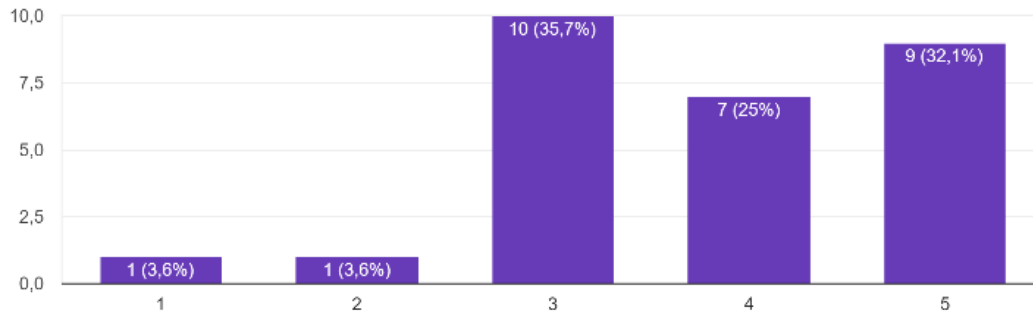


Figure 6-7. Results on PU2

Figure 6-7 presents the students' answers to PU2. 57.1% consider that the game could make it easier for them to study Computational Thinking concepts and practices, while 7.2% of students answered that they disagree/strongly disagree.

PU3. The prompts the game provide me were enough to help me solve the levels.

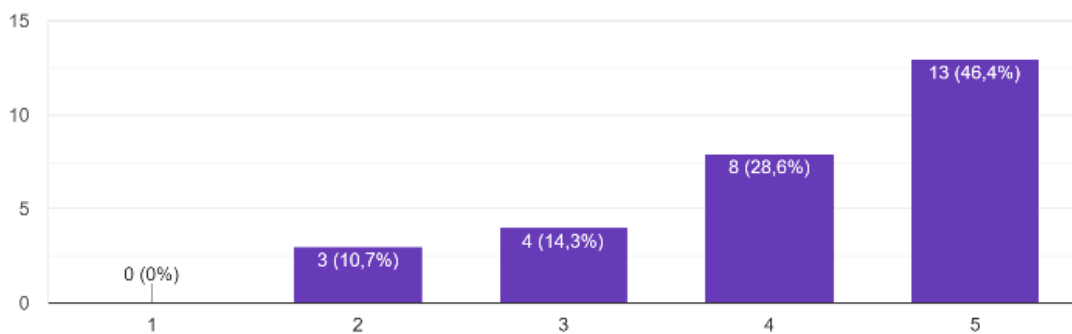


Figure 6-8. Results on PU3

PU4. The prompts the game provide me were useful to help me solve the levels.

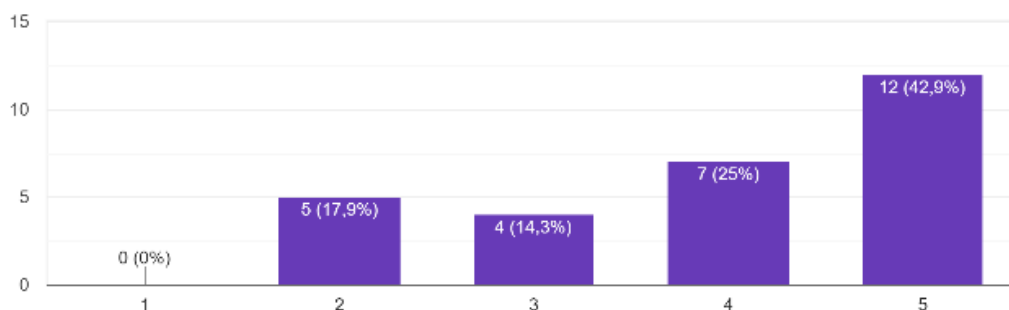


Figure 6-9. Results on PU4

PU5. The prompts the game helped me understand the basic concepts of programming and Computational Thinking.

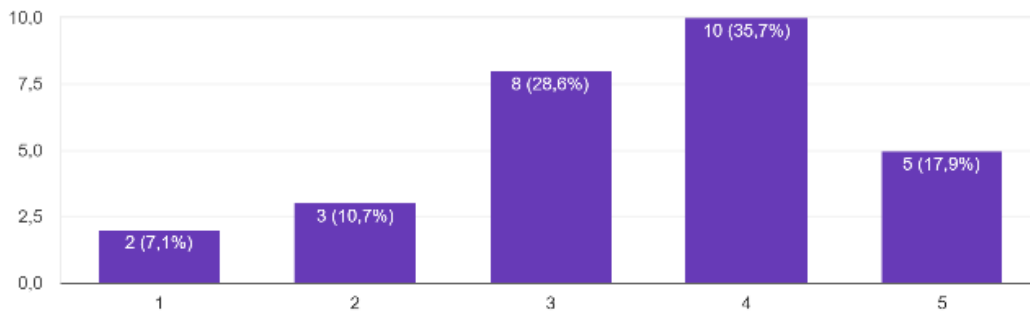


Figure 6-10. Results on PU5

Figure 6-8, Figure 6-9 and Figure 6-10 present the students' answers to PU3, PU4 and PU5 respectively. 75% of students answered that the prompts provided were enough to help them solved the levels. 67.9% found them useful and 53.6% found that the prompts helped them understand Computational Thinking concepts and practices. While only 17.1% stated that they disagree/strongly disagree that the game helped to understand the basic programming and Computational Thinking concepts.

6.4.4 Attitude (AT)

AT1. Studying Computational Thinking and programming through games such as aMazeD is a good idea.

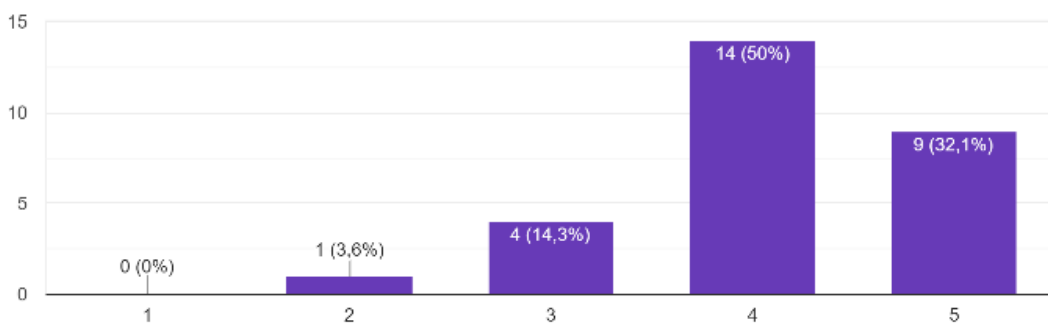


Figure 6-11. Results on AT1

Figure 6-11 presents the students' answers to AT1. 82.1% has a positive attitude towards learning Computational Thinking through games such as aMazeD. While only 3.6% express a negative attitude.

AT2. I'm positive about programming and computational thinking games.

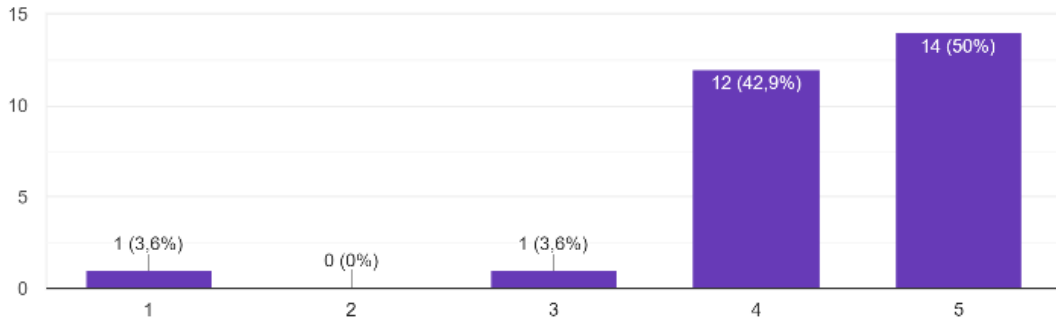


Figure 6-12. Results on AT2

Figure 6-12 presents the students' answers to AT2. 92.9% has a positive attitude towards Computational Thinking games, while 3.6% express a negative attitude.

6.4.5 Accessibility (AC)

AC. I have no difficulty accessing and using the aMazeD programming and Computational Thinking game

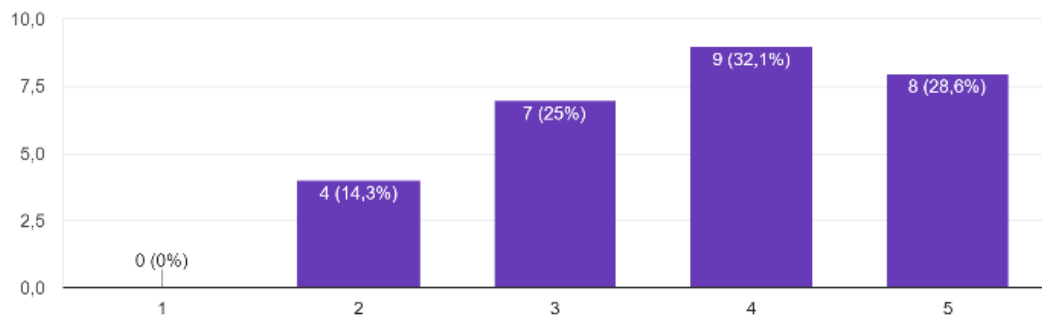


Figure 6-13. Results on AC

Figure 6-13 presents the students' answers to AC. 60.7% had no difficulty in using the aMazeD Computational Thinking game, while 14.3% had difficulties. It is possible that the question wasn't clear enough and students answered in regard the level of difficulty

of the game and not the difficulty in using the game. We base this assumption on the fact that no student reported having difficulty in using the game in the open-ended question, with some students commenting on how challenging/difficult the game was.

6.4.6 Overall experience

Students were asked to answer the following open-ended question: “Write a few words about your experience of playing aMazeD. What did you like or dislike? What impressed you?”. 25 students answered this open-ended question while three left it blank. We coded their answers into two themes: Game overall and Game experience in relation to Computational Thinking and programming.

Regarding how students perceived the game, students generally found the game nice, interesting and fun. 11 students stated that the game was “nice”/ “very nice” / “interesting” / “fun” / “challenging”.

Three students focused on the ease of use of the game. For example, one student stated that “The game is very well designed and easy to use.”

Three students focused on the prompts:

- Student1: “I loved playing this game because of its ease of use. I was impressed by how helpful the tips were.”
- Student2: “This is my second time doing programming, and the instructions given to us helped me to solve them [the levels] more easily.”
- Student3: “I really liked the logic of the game. Also, the prompts were very interesting, although on most levels I did not need them. In addition, the environment was very friendly, simple and convenient. I have only a small objection to a very small detail: the "reset" button could have a repeat icon rather than an “X”. Also, the submit button could have a tick for icon.

The majority of the students also perceived the game as effective on learning Computational Thinking and programming. This is supported by the following quotes:

- Student4: “It was a really nice experience. The game helps in thinking and creativity.”
- Student5: “I liked that it helped me understand Computational Thinking a little bit.”
- Student6:” I liked it and it helped me to understand some things.”
- Student7: “The game was interesting to get acquainted with the programming.”
- Student8: “I quite liked it because it is a fun way to learn things about programming”.
- Student9: “The thought process helps you understand Computational Thinking concepts.”

Finally, only two students express moderate or negative statements about the game.

- Student10: “Although I did not find it very useful it was quite interesting.”
- Student11: “I didn’t like it.”

6.5 Discussion

In this study we design and evaluate a Scaffolding Computational Thinking game. We present the aMazeD game that provides Computational Thinking activities to students and includes scaffolding features. We also present the results of the evaluation of the game and its features. The aMazeD Computational Thinking game is developed to cover Computational Thinking core concepts and practices and to support scaffolding. The scaffolding features include a) the provision of semi-finished or incorrect solutions, b) the provision of explanations for the basic Computational Thinking concepts and c) the provision of prompts that explain the logic behind the solution of the game.

The results of the evaluation regarding ease of use, usefulness, attitude, accessibility and overall experience are promising. Specifically, students seem to consider aMazeD and similar games as easy to use and accessible. What is also important is that students are in general positive to Computational Thinking games. The results in questions regarding how students perceive usefulness of the game indicate that Computational

Thinking and programming games could help students develop Computational Thinking. This is constant with prior research e.g. (Zhao & Shute, 2019; Karakasis & Xinogalos, 2020) that found that programming games could be effectively utilized to help students develop their Computational Thinking. It is characteristic that a high percentage of 92% believe that the game could improve their Computational Thinking. Students also found scaffolding features and specifically prompts useful for solving the game and effective in learning Computational Thinking. This is reflected in their answers to the open-ended question where they evaluate the game and their experience as a whole. Almost all the comments are extremely positive, focusing on both the ease of use of the game and the effectiveness of its scaffolding features.

6.6 Summary

This chapter presents the method and results of the third phase of this dissertation that involves the design and evaluation of a Scaffolding Computational Thinking game. For this purpose, a Computational Thinking game with scaffolding features, was designed and evaluated by 28 middle school students. The study adopts a survey research approach. The results regarding ease of use, usefulness, attitude, accessibility and overall experience of the scaffolding game are promising. Specifically, students found scaffolding features useful for solving the game and effective in learning Computational Thinking.

7 The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking

7.1 Introduction

This chapter presents an experimental study that aims to investigate the effect of scaffolding programming games on the development of middle students' Computational Thinking.

The remainder of this Chapter is organised as follows: Section 7.2 presents literature review of scaffolding and attitudes investigated in Computational Thinking studies. This section has been added to provide basic concepts and previous work on scaffolding and attitudes towards programming for better understanding of the background theory of the study described in the chapter. Section 7.3 presents the design of the experimental study. Sections 7.4-7.11 present the effects of scaffolding programming games on the development of middle students' Computational Thinking. Section 7.12 further discusses the chapters results. Section 7.13 presents a summary of the chapter.

7.2 Related Work

In the process of teaching and learning Computational Thinking, learning strategies play an important role. Efforts have been made to investigate several pedagogies and learning strategies for teaching Computational Thinking. Among them, game-based learning and scaffolding are widely adopted (Hsu et al., 2018). Game-based approaches can increase student motivation, address their disengagement, and foster the acquisition of Computational Thinking (Weintrop et al., 2016). Thus, they are exploited in several studies (e.g., de Souza et al., 2019; Garneli & Chorianopoulos, 2018, 2019; Israel-Fishelson & Hershkovitz, 2020; Zhao & Shute, 2019). In addition to game-based learning, scaffolding is proposed (Repenning et al., 2015) to increase motivation and student participation in Computational Thinking. Studies also (e.g., Angeli & Valanides, 2020) reveal that there is a need to scaffold students' learning during their engagement with Computational Thinking. According to Denner et al. (2012), without proper guidance students face

significant challenges in developing Computational Thinking skills. Scaffolding helps students better understand Computational Thinking concepts, which they would not be able to assimilate if left alone to experiment in a programming environment (Grover et al., 2015). The aforementioned efforts highlight the importance of feedback and guidance strategies in Computational Thinking approaches. However, more research is needed on how the absence versus presence of scaffolding strategies could affect students' cognitive Computational Thinking learning gains.

Technologies and tools are also important. Thus, researchers focus on the development of tools specific to support Computational Thinking learning through programming. Sengupta et al. (2013) developed the CTSiM (Computational Thinking in Simulation and Modelling) tool. CTSiM is a visual programming environment that includes a modelling environment and supports low-threshold, high-ceiling, algorithm visualization, scaffolding and constructivist learning activities. The second version of CTSiM is developed to provide students with adaptive scaffolding based on modelling learner's domain knowledge, cognitive skills and interests (Basu et al., 2017). Weintrop et al. (2016) developed a constructionist video game aiming to foster Computational Thinking. RobotBuilder features a block-based programming language to allow students to construct their game strategies. Clark and Sengupta (2019) developed the SURGE: Gameblox, a Disciplinary-Integrated Game (DIG). SURGE: Gameblox exploits formal representations (such as scientific graphs) and agent-based game programming in a collaborative environment targeting on promoting Computational Thinking. Although the aforementioned tools have been developed to include features that support specific learning strategies, more empirical research that aims to investigate the relationship between tools, learning strategies and Computational Thinking development (Tikva & Tambouris, 2021b) is needed.

In addition to learning strategies and tools, research studies are interested in how various factors influence the acquisition of Computational Thinking. Research (e.g., Kong et al., 2018) has focused on exploring students' attitudes towards programming in the context of Computational Thinking. Particular interest has been paid on how several Computational Thinking interventions could improve students' attitudes towards programming. For example, Cetin (2016) explored the effect of a Scratch-based

intervention on students' attitudes towards programming. However, studies that explore the relationship between attitudes towards programming and Computational Thinking acquisition are scarce (Sun et al., 2022).

7.2.1 Scaffolding strategies in Computational Thinking research

Scaffolding strategies including instructional scaffolding, adaptive, peer-, resource-scaffolding support/guidance, feedback and prompts have been explored in several studies focusing on the development of Computational Thinking (Tikva & Tambouris, 2021a). Chevalier et al. (2022) investigated the role of different types of guidance and feedback in the development of Computational Thinking. To this end, they designed an experimental study to investigate which of these methods fosters students' Computational Thinking. They explored four experimental conditions for the different combinations of with/without guidance and immediate/delayed feedback strategies. Their results support that delayed feedback could be an effective intervention method for Computational Thinking development. Angeli and Valanides (2020) investigated the impact of two scaffolding techniques, designed with gender differences into consideration. To this end, students were randomly assigned to two groups, each following a different type of scaffolding. Their findings show that both sexes benefited from both scaffolding techniques, while each gender benefited more from a different scaffolding technique. Chen et al. (2021) designed a quasi-experimental study to investigate the effects of scaffolding prompts on students' Computational Thinking. Students were assigned to three groups, each of which received cognitive prompts, metacognitive prompts and combination of cognitive and metacognitive prompts respectively. Their findings support that metacognitive scaffolding prompts could be an effective strategy to foster student's Computational Thinking. In the same line, Atmatzidou et al. (2018) explored the effects of different types of guidance (minimal vs strong) on students' metacognitive and problem-solving skills. The findings of their quasi-experimental study support that strong guidance could have a positive impact on students' metacognitive and problem-solving skills.

7.2.2 Attitudes towards programming/Computer Science in Computational Thinking research

Attitudes towards programming and Computer Science (CS) are of interest to Computational Thinking studies. Attitudes towards programming are explored under two major research questions: a) To what extent do specific interventions impact students' attitudes towards programming/CS? and b) To what extent students' attitudes towards programming/CS affect their Computational Thinking? For example, Zhao and Shute (2019) measure attitudes toward CS based on a survey that includes questions about how students perceive computers such as “Computers are fun” and “Computing jobs are boring”. Subsequently they explored if playing a programming video game could have an impact on students' attitudes, finding no statistically significant differences in students' attitudes before and after the intervention. They point out that the short duration of the intervention may have played a role in this outcome. In the same line, Cetin (2016) explored the effects of a Scratch-based instruction on participants' attitudes towards programming, finding no statistically significant effect. They suggest that this could be attributed to the limited duration of treatment, the participants' already high attitudes and satisfaction with the quality of teaching.

Other studies focus on how students' attitudes towards programming could affect Computational Thinking acquisition. For example, Sun et al. (2022) define programming attitude based on a framework that includes the elements of programming self-efficacy, programming utility, social needs, perceptions of programmers, and programming interest. Their results support that students' attitudes towards programming could impact their Computational Thinking, indicating them as an important factor in Computational Thinking development. Kong et al. (2018) define programming empowerment as a Computational Thinking perspective. They explore whether interest in programming and attitude towards collaboration are related to programming empowerment. Their results suggest that interest in programming could affect the acquisition of programming empowerment.

Despite the interest in attitudes towards programming/CS, there is no unanimously accepted definition by researchers. Computational Thinking studies explore various attitudes, while focusing on developing scales for them (e.g., Cetin & Ozden, 2015). Table

7-1 presents attitudes that appear repeatedly in the literature. In the context of this study, attitudes towards programming consist of the following three (3) dimensions: programming self-efficacy, interest in programming and programming meaningfulness.

Table 7-1. Attitudes towards programming/CS found in the literature

Attitude		Scale item example	Study
Confidence/ Self-efficacy	programming self-efficacy	I am good at programming (Kong et al., 2018)	Kukul et al., 2017 Kong et al., 2018 Durak et al., 2019
	CS self-efficacy	I feel confident about my ability to use computers (Werner et al., 2012)	Werner et al., 2012 Román-González et al., 2018
	coding confidence	I am good at coding (Mason & Rich, 2020)	Mason & Rich, 2020
	programming confidence	I am confident to learn programming (Sun et al., 2022)	Sun et al., 2022
Interest	interest in programming	I think the content of programming is fun (Kong et al., 2018)	Kong et al. 2018 Sun et al., 2022
	coding interest	Solving coding problems seems fun (Mason & Rich, 2020)	Mason & Rich, 2020
	programming	Programming is useful to me (Kong et	Kong et al.,

Meaningfulness/Utility	meaningfulness	al., 2018)	2018
	coding utility	Knowing how to code will help me to create useful things (Mason & Rich, 2020)	Mason & Rich, 2020
	programming utility	Learning programming is very useful (Sun et al, 2022)	Sun et al, 2022
Social influence/needs		My parents think coding is important (Mason & Rich, 2020)	Mason & Rich, 2020 Sun et al, 2022
Perception of coders/ programmers		I think kids who can code spend less time outdoors than other kids (Sun et al, 2022)	Mason & Rich, 2020 Sun et al, 2022

7.3 Study design

7.3.1 Study goal and research questions

This study aims to investigate the effect of scaffolding programming games on the development of middle students' Computational Thinking (CT). An additional goal is to investigate the effect of middle school students' attitudes towards programming in their Computational Thinking development. For this purpose, the "aMazeD" (Chapter 6) was utilized. The scaffolding game is aligned with CT concepts and practices included in Brennan's and Resnik's (2012) framework. In particular, we explore how the presence of scaffolding features affect the acquisition of students' Computational Thinking. In addition, herein we investigate the effect of students' attitudes towards programming on their Computational Thinking improvement.

The following research questions are posed:

RQ1. Does aMazeD have a positive impact on middle school students' CT development?

RQ2. Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD version without scaffolding?

RQ3. Do attitudes towards programming have an impact on middle school students' CT?

RQ4. Do attitudes towards programming have an impact on middle school students' CT improvement?

7.3.2 Research design

In order to address the study goal, we conducted an experimental study. Ethical approval from the university ethical committee of the authors' university was obtained. In addition, all students' parents were informed and gave their consent to participate in the study. Participants were 57 students in seventh, eighth and ninth grade. From them, 29 students were randomly assigned to the experimental group where a scaffolding version of the programming game was used as the learning approach, while the rest 28 students were assigned to the control group where a version of the programming game that did not include scaffolding features was used. In order to prevent potential influence of different teachers on the outcome of the study, all students were taught by the same teacher using the same technical equipment regardless of which group they belonged to. The experiment was conducted in three phases and lasted three weeks. In the first phase, students were asked to complete a pre-test for measuring their Computational Thinking and a questionnaire measuring their attitudes towards programming. Both the pre-test and the questionnaire lasted 45 minutes. Students completed the pre-test and the questionnaire on two different days. In the second phase of the experiment, students participated in a 45-minute intervention where they were introduced to Computational Thinking through the two versions of the programming game, depending on the group they belonged to. During the intervention, students encountered Computational Thinking concepts such as sequence, loops, conditionals and Computational Thinking practices such as testing and debugging and being incremental and iterative. Log files from the game were also collected. In the last phase, students completed a post-test for measuring their Computational Thinking which lasted 45 minutes.

7.3.3 Intervention instrument

The “aMazeD” scaffolding programming game presented in the previous Chapter was utilized as the tool through which students were introduced to Computational Thinking.

7.3.4 Data collection

In this study, we measured students’ pre-intervention and post-intervention Computational Thinking using the Computational Thinking Test (CTt). The CTt was developed and validated by Román-González et al. (2017). A translated version of the CTt that authors shared with us, is presented in Appendix C. The CTt is a direct assessment method that is widely accepted as a reliable way to measure Computational Thinking. It consists of 28 multiple choice items. Questions are presented using the interface of Maze or Canvas and the answers are presented as visual arrows or blocks.

We also collected the aMazeD log files that include the following information for each student: a) the success or failure in each level and b) the code submitted for each level.

An instrument for measuring attitudes towards programming was adapted from Kong (2018). We used the following three constructs of the aforementioned instrument translated in the students’ native language: programming meaningfulness, programming self-efficacy and interest in programming to measure students’ attitudes towards programming. The scale consists of 13 items and students were asked to indicate their level of agreement with each item on a 5-point Likert scale (1=Strongly agree; 5=Strongly disagree).

7.3.5 Study Limitations

This study has some limitations including the small sample size and the short duration of the intervention. A longer duration could provide more insights on students’ learning gains. In addition, we based our analysis only on tests, questionnaires and logs. Including interviews and video recording could have provided a more holistic understanding of students’ CT development. The inclusion of students from a single school could be also considered as a limitation of the study

7.4 Demographics

57 students whose parents gave their consent to participate in the study were randomly assigned to the control and experimental group. There were 5 students from the control group and 7 from the experimental group who were absent either during the completion of the tests or during the intervention. This resulted in a final sample of 45 students, of whom 23 belong to the control group and the rest 22 to the experimental group. The distribution of students by grade and gender is shown in Table 7-2. Among participants, 23 (51%) students were male and 22 (49%) were female. 13 (29%) were in 7th grade, 21 (47%) were in 8th grade and 11 (24%) were in 9th grade.

Table 7-2. Distribution of participants by grade and gender

		Grade			Gender	
		7th	8th	9th	Male	Female
Version	Non-Scaffolding	7	10	6	14	9
	Percentage in the non-scaffolding group	30.4%	43.5%	26.1%	60.9%	39.1%
	Scaffolding	6	11	5	9	13
	Percentage in the scaffolding group	27.3%	50%	22.7%	40.9%	59.1%

7.5 CTt

CTt (Román-González et al., 2017) was employed to measure CT pre-intervention and post-intervention scores. For each item we assigned 1 if it was correct and 0 if it was incorrect. The score for each test ranged from 0 to 28. The scale had an acceptable level of internal consistency, as determined by a Cronbach's alpha of .763 reported in the pre-intervention data and an acceptable level of internal consistency as determined by a Cronbach's alpha of .803 reported in the post-intervention data.

7.6 Analytics

We calculated the overall game score for each student based on aMazeD game logs. For each level we assigned 1 if it was successfully completed and 0 otherwise. The overall

game score for each student ranged from 0 to 10. The Cronbach's alpha coefficient was 0.753. We also calculated the following scores based on the inspection of the submitted code:

- Conditional-Level and Loop-Level score. We assigned 1 for each successfully completed level belonging to the “Conditionals” concept (Table 1) and 0 otherwise. The overall Conditional-Level score for each student ranged from 0 to 3. Accordingly, we assigned 1 for each successfully completed level belonging to the “Loops” concept (Table1) and 0 otherwise. The overall Loop-Level score for each student ranged from 0 to 8.
- Conditional-Use and Loop-Use score. We assigned 1 if the submitted code contained Conditionals for each correctly completed level belonging to the “Conditionals” concept and 0 otherwise. The overall Conditional-Use score for each student ranged from 0 to 3. Accordingly, we assigned 1 if the submitted code contained Loops for each correctly completed level belonging to the “Loops” concept and 0 otherwise. The overall Loop-Use score for each student ranged from 0 to 8.
- Conditional-Ratio and Loop-Ratio. We calculated the Conditional-Ratio as the ratio between Conditional-Use score and Conditional-Level Score and the Loop-Ratio as the ratio between Loop-Use score and Loop-Level score.

7.7 Scale of Attitudes towards Programming

A scale adapted from Kong (2018), was used to measure student’s attitudes towards programming. The scale consisted of 13 items 5-point Likert scale, (1 = Strongly agree and 5 = Strongly disagree). The score of each student was calculated as the sum of the 13 items and ranged from 13 to 65. 40 of the participants were filled in the attitudes towards programming scale. The scale had a high level of internal consistency, as determined by a Cronbach's alpha of 0.948 (Table 7-3). We classified the participants into three groups based on their percentile value in the scale score distribution: Low-attitudes towards programming students (n=13), Moderate-attitudes towards programming (n=14) and High-attitudes towards programming students (n=13).

Table 7-3. Internal consistency of the scale of Attitudes towards Programming

Construct	Number of items	Cronbach's alpha
programming meaningfulness	4	0.921
programming self-efficacy	5	0.912
interest in programming	4	0.900
Entire scale	13	0.948

7.8 Does aMazeD have a positive impact on middle school students' CT development?

The first research question was, “Does aMazeD have a positive impact on middle school students' CT development?” Our hypothesis was that aMazeD would have a positive impact on middle school students' CT development. A paired-samples t-test was used to determine whether there was a statistically significant mean difference between the pre-intervention CT scores and the post-intervention CT scores of the students. No outliers were detected. The assumption of normality was not violated, as assessed by Shapiro-Wilk's test ($p = .612$). We found a significant mean increase of 3.933, 95% CI [3.097, 4.769], $t(44)=9.481, p<.001$ between pre-intervention and post-intervention CT scores, with a large effect size (Cohen's $d=1.413$). Students CT post-intervention scores were higher ($M=19.333, SD=4.772$) compared to their CT pre-intervention scores ($M=15.4, SD=4.653$). This result supports our hypothesis that aMazeD would have a positive impact on students' CT development.

7.9 Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD version without scaffolding features?

The second research question was “Does aMazeD with scaffolding features have a greater impact on middle school students' CT development than the aMazeD without scaffolding?”. Our hypothesis was that the scaffolding version of aMazeD would have a

greater impact on students' CT development. CT pre-scores and post-scores were measured by the CTt (Román-González et al., 2017). An independent t-test showed that the mean of the pre-test CT scores of the scaffolding group was not significantly higher ($M=15.727$, $SD=4.442$) than that of the non-scaffolding group ($M=15.087$, $SD=4.926$); $t(43) = -.457$, $p=.650$. Thus, we can conclude that the two groups were equivalent in terms of students' CT scores prior to the intervention. An ANCOVA was run to determine the effect of the scaffolding version of the game on post-intervention CT scores after controlling for pre-intervention CT scores. There was a linear relationship between pre-intervention CT scores and post-intervention CT scores for each group, as assessed by visual inspection of a scatter plot. There was homogeneity of regression slopes as the interaction term was not statistically significant, $F(1,41) = .180$, $p = .673$. Standardized residuals for the interventions and for the overall model were normally distributed, as assessed by Shapiro-Wilk's test ($p > .05$). There was homoscedasticity and homogeneity of variances, as assessed by visual inspection of a scatterplot and Levene's test of homogeneity of variance ($p = .911$), respectively. There were no outliers in the data, as assessed by no cases with standardized residuals greater than ± 3 standard deviations. After adjustment for pre-intervention CT scores, there was a statistically significant difference in post-intervention CT scores between the scaffolding and the non-scaffolding group, $F(1,42) = 5.657$, $p = .022$.

We further analyze students' log files. Mann-Whitney U test was run to determine if there were differences in Conditional-Use scores between the non-scaffolding and scaffolding group. Distributions of the Conditional-Use scores for the two groups were not similar, as assessed by visual inspection. Conditional-Use scores for the scaffolding group (mean rank = 29.30) were statistically significantly higher than for the non-scaffolding group (mean rank = 16.98), $U = 391.5$, $z = 3.409$, $p = .001$. Respectively, Mann-Whitney U test was run to determine if there were differences in Loop-Use Score between the non-scaffolding and scaffolding group. Distributions of the Loop-Use Scores for the two groups were not similar, as assessed by visual inspection. Loop-Use scores for the scaffolding group (mean rank = 30.27) were statistically significantly higher than for the non-scaffolding group (mean rank = 16.04), $U = 413$, $z = 3.695$, $p < .001$.

7.10 Do attitudes towards programming have an impact on students' CT?

The third research question was “Do attitudes towards programming have an impact on middle school students' CT?”. Our hypothesis was that positive attitudes towards programming would have a greater impact on students' CT scores. A one-way ANOVA was conducted to determine if the students' CT pre-test scores were different for the low/moderate/high attitudes groups. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro-Wilk test ($p > .05$); and there was homogeneity of variances, as assessed by Levene's test of homogeneity of variances ($p = .818$). CT pre-test score increased from low ($M=13.769$, $SD=4.902$) to moderate ($M=15.429$, $SD=4.327$) to high ($M=17.154$, $SD=4.793$) attitudes group, in that order, but the differences between attitudes groups was not statistically significant, $F(2,37) = 1.706$, $p = .196$. This result does not support the hypothesis that student's attitudes towards programming would have an impact on middle school students' CT.

7.11 Do attitudes towards programming have an impact on students' CT improvement?

The fourth research question was “Do attitudes towards programming have an impact on students' CT improvement?”. Our hypothesis was that attitudes towards programming would have an impact on students' CT development. A one-way ANOVA was conducted to determine if the changes in students' CT scores were different for the low/moderate/high attitudes groups. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro-Wilk test ($p > .05$); and there was homogeneity of variances, as assessed by Levene's test of homogeneity of variances ($p = .113$). Changes in CT scores increased from moderate ($M = 3.143$, $SD=3.348$), to high ($M=3.539$, $SD=1.808$), to low ($M=4.462$, $SD=2.817$) attitudes group, but the differences were not statistically significant, $F(2,37) = .807$, $p = .454$. This result does not support the hypothesis that student's attitudes towards programming would have an impact on middle school students' CT development.

7.12 Discussion

Our first hypothesis was that aMazeD would have a positive impact on middle school students' CT. Data analysis and results seem to support this hypothesis. Participants significantly improved their CT scores at the CTt after playing the aMazeD. This is consistent with prior research showed that playing programming games could improve students' Computational Thinking (e.g., Hooshyar et al., 2021; Zhao & Shute, 2019). However, since this is a one-group pretest-posttest design, it cannot be excluded that the differences between the pre-test and post-test are due to threats such as maturation (Fraenkel et al., 2012).

The second hypothesis was that aMazeD with scaffolding features would have a greater impact on middle school students' CT than the aMazeD version without scaffolding features. Both groups experienced an improvement in their post-intervention CT scores, but students who played the scaffolding version of the game had significantly higher CT post-scores (Table 7-4). Furthermore, students in the scaffolding group not only did better on the post-test, but they had significantly higher Conditional-Use and Loop-Use scores (Table 7-5). The code they submitted to the game was of higher quality and included the use of Conditionals and Loops. It is indicative that students in the scaffolding group who used conditionals in all successful levels belonging to the "Conditional Concept" concept amount to 18 out of 22 compared to 6 out of 23 students in the non-scaffolding group. Respectively, students in the scaffolding group who used loops in all successful levels belonging to the "Loop Concept" amount to 18 out of 22 compared to 4 out of 23 students in the non-scaffolding group. These results suggest that scaffolding could be an effective learning technique for developing students' CT and help them understand the core concepts of CT such as Conditionals and Loops. Prior research also shows results regarding the relationship between scaffolding and CT development. Studies conclude that scaffolding could have a positive impact on CT development. Specifically, Chen et al. (2021) findings of their quasi-experimental study revealed that metacognitive prompts significantly improved students' CT outcomes. In the same line, Angeli and Valanides (2020) found that students who participated in their study benefited from the scaffolding techniques used. Furthermore, Chevalier et al. (2022) found that students in their study benefited from guidance and feedback learning methods.

Table 7-4. Computational Thinking pre-scores and post-scores means by game version

Game Version	Means of Pre-intervention Scores	Means of Post-intervention Scores	Means of CT scores changes
Scaffolding version	15.727	20.546	4.818
Non-Scaffolding version	15.087	18.174	3.087

Table 7-5. Computational Thinking Conditional-Level, Loop-Level, Conditional-Use, Loop-Use scores, Conditional-Ratio and Loop-Ratio means by game version

Game Version	Means of Conditional-Level Scores [0-3]	Means of Loop-Level Scores [0-8]	Means of Conditional-Use Scores [0-3]	Means of Loop-Use Scores [0-8]	Means of Conditional-Ratio	Means of Loop-Ratio
Scaffolding version	2.86	6.05	2.50	5.36	0.871	0.878
Non-Scaffolding version	2.57	4.65	1.13	2.22	0.384	0.409

The third hypothesis was that attitudes towards programming would have an impact on students' CT scores. No significant differences were found between the three groups (low/moderate/high) in the results of students' CT pre-tests. Although students' pre-test scores were very similar in general, as shown in Figure 7-1, the students of the low attitudes group were less successful than students in the moderate and high attitudes group. Previous studies indicate that Computational Thinking is related with attitudes towards programming (Sun et al., 2022) and suggest that interest in programming could be an important factor in the acquisition of CT (Kong et al., 2018), proposing interest-driven strategies for CT teaching and learning (Kong, 2016).

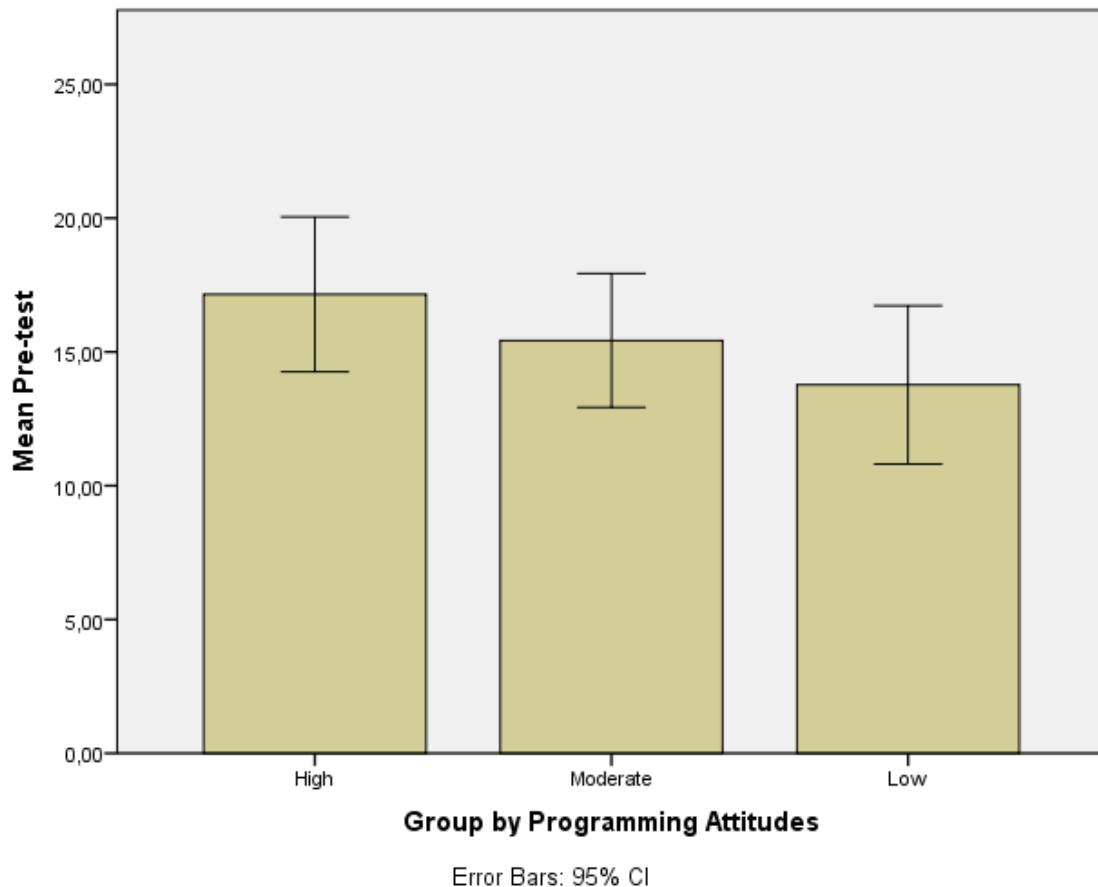


Figure 7-1. Means of pre-tests scores by attitudes towards programming group

The fourth hypothesis was that attitudes towards programming would have an impact on students' CT development. Although this hypothesis was not confirmed as no significant differences were found between the three groups (low/moderate/high) in students' CT improvement, the descriptive statistical analysis reveals interesting results. As shown in Table 7-6, changes in students' CT scores for the non-scaffolding version increase from low ($M= 1.600$, $SD=.872$) to moderate ($M=2.556$, $SD=1.069$), to high attitudes group ($M=4.000$, $SD=5.35$) (Figure 7-2). This result is consistent with other studies (Sun et al., 2022) which have shown that students with negative attitudes towards programming may find it more difficult to develop their Computational Thinking than students with positive attitudes towards programming. Results indicate that students are struggling to develop their Computational Thinking skills when they are not provided with an appropriate learning strategy. This is in line with previous studies which suggest that students face great difficulties without proper guidance (Denner et al., 2012). However, this is not the case for students that experienced the scaffolding version. Changes in

students' CT scores in the scaffolding version increase from high ($M=3.000$, $SD=.894$) to moderate ($M=4.200$, $SD=1.655$) to low ($M=6.250$, $SD=.491$) attitudes group (Figure 7-3). This result could have important implications in the design of appropriate learning interventions regarding the choice of the learning strategies in relation to students' attitudes towards programming. Results suggest that students with low and moderate attitudes towards programming tend to benefit more from the scaffolding strategy than students with higher attitudes towards programming. The provision of scaffolding through semi-finished programs and prompts could engage students who tend to have low interest in programming and low programming self-efficacy, by reducing difficulty levels and providing effective supplies for developing Computational Thinking.

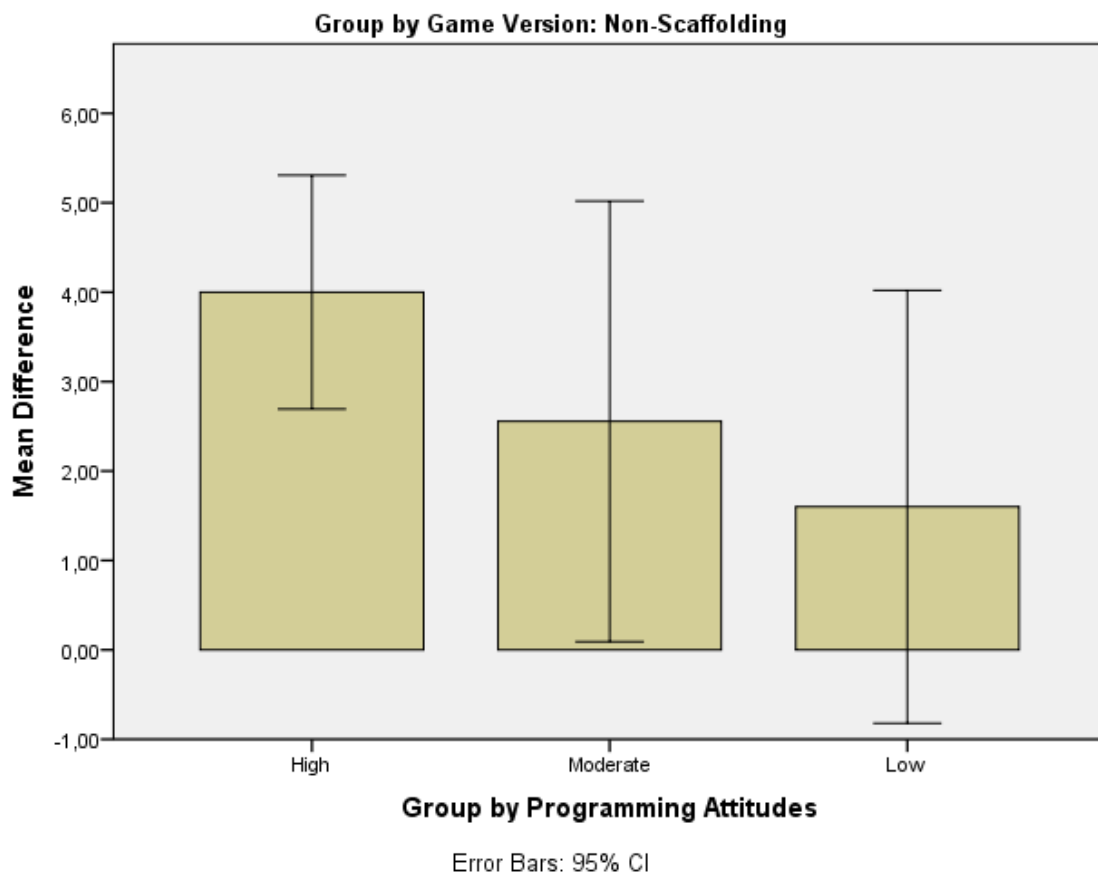


Figure 7-2. Means of score changes by attitudes towards programming group for the non-scaffolding group

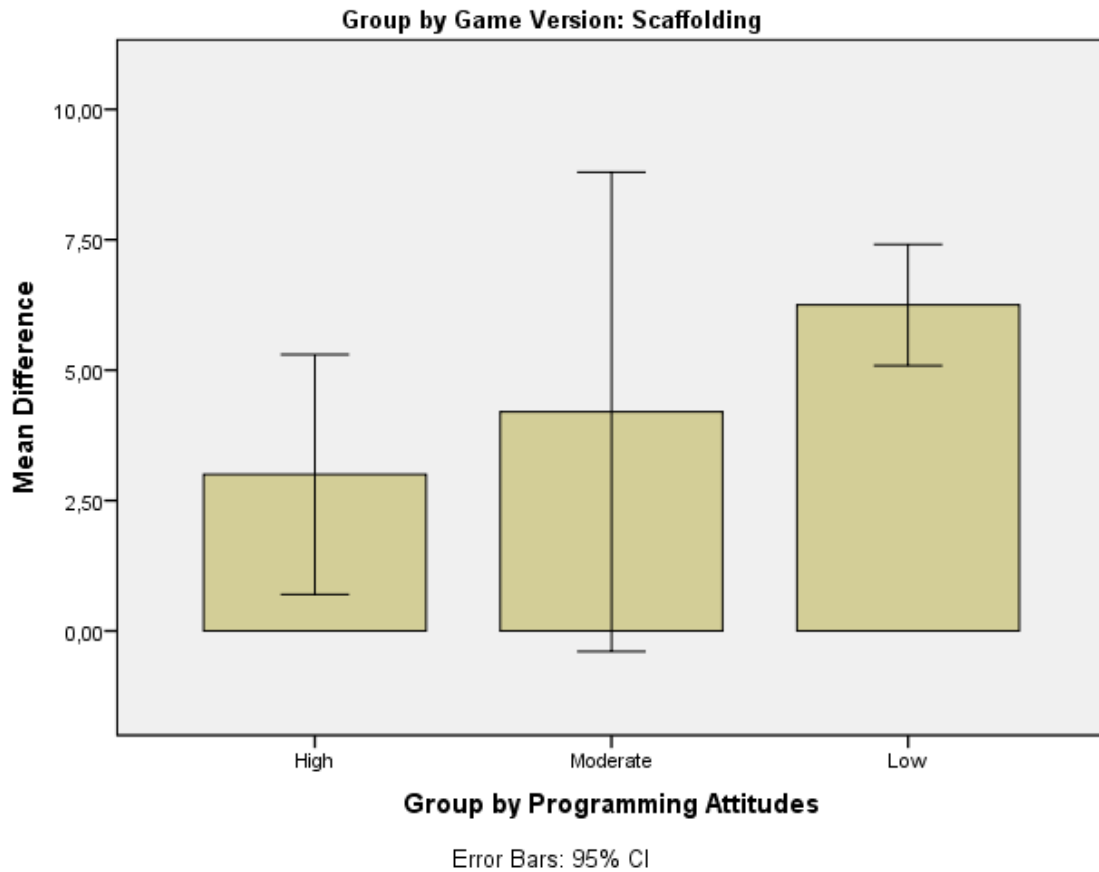


Figure 7-3. Means of score changes by attitudes towards programming group for the scaffolding group

Table 7-6. Computational Thinking changes in pre-scores and post-scores means by game version and attitudes towards programming group

Game Version	Attitudes towards programming Group	Means of Change in CT Scores
Non-scaffolding version	High	4.000
	Moderate	2.556
	Low	1.600
Scaffolding version	High	3.000
	Moderate	4.200
	Low	6.250

The implication of these findings is important, as they provide support that scaffolding in computational thinking games could be an effective strategy for teaching and learning computational thinking to middle school students fostering a deeper understanding of Computational Thinking concepts. In addition, when it comes to students' attitudes towards programming, students who perceive programming as less meaningful, less interesting and have lower programming self-efficacy could particularly benefit from scaffolding aspects in programming games.

7.13 Summary

This chapter presents the method and results of the fourth phase of this dissertation that involves the investigation of the effects of a) scaffolding programming games and b) attitudes towards programming, on the development of middle school students' Computational Thinking. To this end, an experimental study was conducted. Students were introduced to CT under two distinct experimental conditions: a scaffolding version of a programming game and a non-scaffolding version of the same game. Results report statistically significant differences between the pre-intervention and post-intervention CT scores for all students and statistically significant improvement in learning outcomes in favor of the scaffolding group. In addition, the study hypothesized that attitudes towards programming would have an impact on students' CT. Although this hypothesis has not been confirmed, the results suggest that students who have a less positive attitude towards programming could particularly benefit from scaffolding aspects in programming games.

8 Conclusions and direction for future research

8.1 Introduction

In the context of this dissertation, we developed a conceptual model of Computational Thinking in K-12 education and extended it to higher education. We also investigated specific instances of the models' CT areas. The research was organized in the following four phases:

Phase 1. Developing a Conceptual Model of Computational Thinking through programming in K-12 education (CTPK-12).

Phase 2. Extending the Computational Thinking through Programming in K-12 Education (CTPK-12) Conceptual Model for Higher Education.

Phase 3. Designing and evaluating of a programming game to study the perceived effects of a certain instance of the CTPK-12 Learning Strategies area.

Phase 4. Using the CTPK-12 model to design an empirical study to investigate certain instances of the Learning Strategies and Factors model's areas.

In this chapter we present the conclusions of these phases. The remainder of this Chapter is organised as follows: Section 8.2 presents the conclusions of Phase 1. Section 8.3 presents the conclusions of Phase 2. Section 8.4 presents the conclusions of Phase 3. Section 8.5 presents the conclusions of Phase 4. Section 8.6 suggests future research. Section 8.7 presents the limitations of the research presented.

8.2 Conclusions Phase 1

In this phase, a conceptual model of CT through programming in K-12 education (CTPK-12) was developed. The proposed model was based on a systematic literature review and the identification of CT Areas and their relationships. CT Areas are determined from the recording of all topics of interest to researchers. CTPK-12 model provides an overall map

of the domain that aids domain understanding and could serve as a basis for future studies as well as facilitate the integration of CT into K-12 educational practices.

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

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

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

Tools area is also one of the major topics investigated in the selected studies. Several studies focus on the development of environments designed specifically to support CT teaching and learning strategies. Although these environments are designed on the basis of CT frameworks, they are not yet widely used in empirical studies or educational practices aimed at developing CT. Instead, they appear only once in the literature in the studies where they are introduced. Therefore, beyond the theoretical basis and technicalities of CT tools, researchers need to consider issues of usability, student

motivation, teacher facilitation through available resources and frameworks, and ease of assessment through built-in automated assessment methods. In addition, future studies should explore the relationship between Tools and CT development providing insights on which tools could better support which CT learning strategies.

Learning Strategies area has gained increasing interest in recent years. However, several of the studies reviewed simply refer to the learning strategies applied without further describing how they were implemented. Focusing on learning strategies, presenting the relevant background and how they are implemented could support a more comprehensive picture of the conditions and context of the proposed CT interventions. Studies could also propose frameworks that support leveraging CT learning strategies. In addition, future studies could explore the relationship between learning strategies and CT development and provide insights on which learning strategies are most suitable for students to acquire which CT elements.

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

Factors area has also been investigated in several of the selected studies. However, some of the results of the studies are contradictory, so it is unclear whether and to what extent these factors lead to higher or lower CT levels. As Angeli & Giannakos (2020) point out, how CT skills, such as abstraction, problem decomposition, and data structures, map to different abilities, grade level, disciplines, gender, and educational level is still missing

from the literature. Further studies in this direction could build clarity about factors that may affect CT acquisition. With regard to how CT could be utilized to motivate underrepresented groups, there are few studies (e.g., Kim & Kim, 2016; Leonard et al., 2018; Pinkard, Martin, & Erete, 2019) specifically aimed at motivating girls and underrepresented minorities. More studies are required to provide evidence of the relationship between factors, learning strategies and tools and provide insights on if and how learning strategies and tools could broaden participation in CT and address challenges related to factors.

8.3 Conclusions Phase 2

The results of this phase indicate that several efforts have been emerged in CT through programming in higher education research recently, although challenges remain in the six areas identified in this review: Knowledge Base, Learning Strategies, Tools, Assessment, Factors and Capacity Building. Future studies should address remaining challenges by basing their design on clear definitions of CT as categorized and described in section 5.4.1.1. The assessment should be based on the recording of CT elements as previously defined in the context of the studies. In addition, it is proposed to integrate direct assessment methods in combination with self-report methods in order to provide a more objective picture of the development of students' CT. The alignment of CT elements and assessment methods could provide a more comprehensive understanding of students' CT development. Future research should also explore how different learning strategies could support CT development. In addition, future research could focus on the development of tools suitable for higher education, which would enable the exploitation of game design strategies. Finally, studies should also focus on the investigation of how factors such as gender, creativity, self-efficacy, motivation may affect CT and how professional development of academic staff could enhance the CT integration in higher education context.

8.4 Conclusions Phase 3

In this phase, we design and evaluate a Scaffolding Computational Thinking game. The game was designed to include scaffolding features and was evaluated by 28 middle school students. The results regarding ease of use, usefulness, attitude, accessibility and overall

experience of the scaffolding game were promising. Specifically, students found scaffolding features useful for solving the game and effective in learning Computational Thinking.

8.5 Conclusions Phase 4

In this phase, we explored the effect of scaffolding programming games on the development of middle school students' Computational Thinking. In addition, herein we explore the effect of students' attitudes towards programming on their Computational Thinking. Students were introduced to Computational Thinking under two distinct experimental conditions: a scaffolding version of a programming game and a non-scaffolding version of the same game. Results reported statistically significant learning gains between the pre-intervention and post-intervention CT scores for all students and statistically significant improvement in learning outcomes in favour of the scaffolding group. Furthermore, students in the scaffolding group not only showed better learning outcomes overall, but also submitted higher quality code in terms of using conditionals and loops during the game. The findings support that scaffolding helps students develop Computational Thinking and deepen their understanding of the related concepts. In addition, the study hypothesized that attitudes towards programming would have an impact on students' Computational Thinking and Computational Thinking development. However, this hypothesis was not confirmed from the results that report a non-statistically significant difference in both cases. Nevertheless, students' Computational Thinking in the non-scaffolding group found to be higher for students with a more positive attitude towards programming. Specifically, students in the high attitudes group had greater learning gains, followed by students in the moderate attitudes group and students in the low attitudes group for the non-scaffolding version of the game. On the other hand, students in the low attitudes group had greater learning gains, followed by students in the moderate attitudes and students in the high attitudes group for the scaffolding version of the game.

8.6 Limitations

The study developed a conceptual model for Computational Thinking and investigated some of its instances, following four research phases. Despite its contributions, the findings

must be considered in light of the limitations of its research phases presented in Sections 4.2.3, 5.2.3, 6.24 and 7.2.5 accordingly.

8.7 Future work

Future research could be organized into the following three objectives:

i) Investigate all model relationships. A full investigation of the relationships of the model could contribute to a better understanding of learning and teaching of Computational Thinking. Specifically, future research could focus on a) the relationship between Tools and CT development and provide insights on which tools could better support which CT learning strategies b) the relationship between learning strategies and CT development and provide insights on which learning strategies are most appropriate for students to acquire which CT elements c) the relationship between capacity building and CT development d) the relationship between factors, learning strategies and tools and provide insights on if and how learning strategies and tools could broaden participation in CT e) the relationship between factors and CT development and build clarity about factors that may affect CT acquisition.

ii) Extend the model. Future research could focus on extending the conceptual model to include other approaches such as unplugged approaches.

iii) Use the model to create course designs. Future research could focus on studying design principles that could lead to structured design courses that are based on the proposed CT conceptual model.

Appendixes

Appendix A

Appendix A. List of selected studies (Chapter 4)

- S1 (CSTA), & (ISTE). (2011). Operational definition of computational thinking. Retrieved from <https://www.iste.org/explore/Solutions/Computational-thinking-for-all>
- S2 Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, Vol. 23, pp. 1501–1514. <https://doi.org/10.1007/s10639-017-9675-1>
- S3 Alfayez, A. A., & Lambert, J. (2019). Exploring Saudi Computer Science Teachers' Conceptual Mastery Level of Computational Thinking Skills. *Computers in the Schools*, Vol. 36, pp. 143–166. <https://doi.org/10.1080/07380569.2019.1639593>
- S4 Allsop, Y. (2019) Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, 19, 30–55. <https://doi.org/10.1016/j.ijcci.2018.10.004>
- S5 Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology and Society*, Vol. 19, pp. 47–57. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85000838214&partnerID=40&md5=3f014c90dafb945e90c9552f5a6ef17f>
- S6 Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, Vol. 75, pp. 661–670. <https://doi.org/10.1016/j.robot.2015.10.008>

- S7 Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. Proceedings - Frontiers in Education Conference, FIE. <https://doi.org/10.1109/FIE.2012.6462365>
- S8 Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54. <https://doi.org/10.1145/1929887.1929905>
- S9 Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, 245–250. <https://doi.org/10.1145/1953163.1953241>
- S10 Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior*, Vol. 80, pp. 412–419. <https://doi.org/10.1016/j.chb.2017.04.058>
- S11 Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a Computational Thinking-based science learning environment. *User Modeling and User-Adapted Interaction*, Vol. 27, pp. 5–53. <https://doi.org/10.1007/s11257-017-9187-0>
- S12 Berland, M., & Wilensky, U. (2015). Comparing Virtual and Physical Robotics Environments for Supporting Complex Systems and Computational Thinking. *Journal of Science Education and Technology*, Vol. 24, pp. 628–647. <https://doi.org/10.1007/s10956-015-9552-x>
- S13 Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, Vol. 72, pp. 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>
- S14 Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian*

Journal of Teacher Education, 42(3), 53–72.

<https://doi.org/10.14221/ajte.2017v42n3.4>

- S15 Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Annual American Educational Research Association Meeting, Vancouver, BC, Canada, 1–25. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf
- S16 Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, Vol. 87, pp. 834–860. <https://doi.org/10.3102/0034654317710096>
- S17 Carlborg, N., Tyrén, M., Heath, C., & Eriksson, E. (2019). The scope of autonomy when teaching computational thinking in primary school. *International Journal of Child-Computer Interaction*, 21, 130–139. <https://doi.org/10.1016/j.ijcci.2019.06.005>
- S18 Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, 17, 93–100. <https://doi.org/10.1016/j.ijcci.2018.06.005>
- S19 Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, Vol. 109, pp. 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- S20 Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, Vol. 62, pp. 563–573. <https://doi.org/10.1007/s11528-018-0292-7>
- S21 Clark, D. B., & Sengupta, P. (2019). Reconceptualizing games for integrating computational thinking and science as practice: collaborative agent-based disciplinarily-integrated games. *Interactive Learning Environments*. <https://doi.org/10.1080/10494820.2019.1636071>

- S22 Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). Education: A future for computing education research. *Communications of the ACM*, 57(11), 34–36. <https://doi.org/10.1145/2668899>
- S23 Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. Retrieved from Computing at Schools. website: <https://community.computingschool.org.uk/resources/2324/single>
- S24 Da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, Vol. 18, pp. 17–39. <https://doi.org/10.15388/infedu.2019.02>
- S25 De Souza, A. A., Barcelos, T. S., Munoz, R., Villarroel, R., & Silva, L. A. (2019). Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access*, Vol. 7, pp. 82848–82866. <https://doi.org/10.1109/ACCESS.2019.2924343>
- S26 Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, 58(1), 240–249. <https://doi.org/10.1016/j.compedu.2011.08.006>
- S27 Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33–39. <https://doi.org/10.1145/2998438>
- S28 Dolgopolas, V., Dagienė, V., Jasutė, E., & Jevsikova, T. (2019). Design science research for computational thinking in constructionist education: A pragmatist perspective. *Problemos*, Vol. 95, pp. 144–159. <https://doi.org/10.15388/Problemos.95.12>
- S29 Durak, H. Y., & Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers and Education*, 116, 191–202. <https://doi.org/10.1016/j.compedu.2017.09.004>

- S30 Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, 10(2), 173–197. <https://doi.org/10.30935/cet.554493>
- S31 Fletcher, G. H. L., & Lu, J. J. (2009). Education: Human computing skills: Rethinking the K-12 experience. *Communications of the ACM*, 52(2), 23–25. <https://doi.org/10.1145/1461928.1461938>
- S32 Fronza, I., El Ioini, N., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education*, Vol. 17. <https://doi.org/10.1145/3055258>
- S33 Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cárdenas, L., Pantano, P., & Bilotta, E. (2019). Lesson planning by computational thinking skills in Italian pre-service teachers. *Informatics in Education*, Vol. 18, pp. 69–104. <https://doi.org/10.15388/infedu.2019.04>
- S34 García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, 80, 407–411. <https://doi.org/10.1016/j.chb.2017.12.005>
- S35 Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: exploring computational thinking through code analysis. *Interactive Learning Environments*, Vol. 26, pp. 386–401. <https://doi.org/10.1080/10494820.2017.1337036>
- S36 Garneli, V., & Chorianopoulos, K. (2019). The effects of video game making within science content on student computational thinking skills and performance. *Interactive Technology and Smart Education*. <https://doi.org/10.1108/ITSE-11-2018-0097>
- S37 Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based

- programming environments. *ACM Transactions on Computing Education*, Vol. 17. <https://doi.org/10.1145/3105910>
- S38 Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, Vol. 42, pp. 38–43. <https://doi.org/10.3102/0013189X12463051>
- S39 Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199–237. <https://doi.org/10.1080/08993408.2015.1033142>
- S40 Günbatar, M. S. (2019). Computational thinking within the context of professional life: Change in CT skill from the viewpoint of teachers. *Education and Information Technologies*, Vol. 24, pp. 2629–2652. <https://doi.org/10.1007/s10639-019-09919-x>
- S41 Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, Vol. 17, pp. 229–244. <https://doi.org/10.15388/infedu.2018.12>
- S42 Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, Vol. 27, pp. 628–644. <https://doi.org/10.1080/10494820.2019.1610451>
- S43 Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, Vol. 126, pp. 296–310. <https://doi.org/10.1016/j.compedu.2018.07.004>
- S44 Ioannidou, A., Bennett, V., Repenning, A., Koh, H., & Basawapatna, A. (2011). Computational Thinking Patterns Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom. Annual Meeting of the American Educational Research Association (AERA), 2. Retrieved from <http://www.agentsheets.com>

- S45 Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, Vol. 82, pp. 263–279. <https://doi.org/10.1016/j.compedu.2014.11.022>
- S46 Israel-Fishelson, R., & Hershkovitz, A. (2019). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research*. <https://doi.org/10.1177/0735633119887187>
- S47 Jeong, Y.-S., & Sung, Y.-H. (2019). The effect of network-based PUMA teaching-learning model on information literacy, computational thinking, and communication skills. *Universal Journal of Educational Research*, Vol. 7, pp. 103–113. <https://doi.org/10.13189/ujer.2019.071512>
- S48 Jun, S., Han, S., & Kim, S. (2017). Effect of design-based learning on improving computational thinking. *Behaviour and Information Technology*, Vol. 36, pp. 43–53. <https://doi.org/10.1080/0144929X.2016.1188415>
- S49 Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, Vol. 59, pp. 26–27. <https://doi.org/10.1145/2955114>
- S50 Kale, U., Akcaoglu, M., Cullen, T., & Goh, D. (2018). Contextual Factors Influencing Access to Teaching Computational Thinking. *Computers in the Schools*, Vol. 35, pp. 69–87. <https://doi.org/10.1080/07380569.2018.1462630>
- S51 Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational What? Relating Computational Thinking to Teaching. *TechTrends*, Vol. 62, pp. 574–584. <https://doi.org/10.1007/s11528-018-0290-9>
- S52 Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal Of Modern Computing*, 4(3), 583–596.

- S53 Kim, Y.-M., & Kim, J.-H. (2016). Application of a software education program developed to improve computational thinking in elementary school girls. *Indian Journal of Science and Technology*, Vol. 9.
<https://doi.org/10.17485/ijst/2016/v9i44/105102>
- S54 Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. *Proceedings - 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2010*, (December), 59–66.
<https://doi.org/10.1109/VLHCC.2010.17>
- S55 Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers and Education*, Vol. 127, pp. 178–189.
<https://doi.org/10.1016/j.compedu.2018.08.026>
- S56 Korkmaz, Ö., & Bai, X. (2019). Adapting computational thinking scale (CTS) for chinese high school students and their thinking scale skills level. *Participatory Educational Research*, 6(1), 10–26. <https://doi.org/10.17275/per.19.2.6.1>
- S57 Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, 72, 558–569. <https://doi.org/10.1016/j.chb.2017.01.005>
- S58 Kukul, V., & Karataş, S. (2019). Computational thinking self-efficacy scale: Development, validity and reliability. *Informatics in Education*, Vol. 18, pp. 151–164. <https://doi.org/10.15388/infedu.2019.07>
- S59 Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education*, Vol. 17, pp. 321–340.
<https://doi.org/10.15388/infedu.2018.17>
- S60 Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing Teachers to Engage Rural Students in Computational Thinking Through Robotics, Game Design, and

Culturally Responsive Teaching. *Journal of Teacher Education*, Vol. 69, pp. 386–407. <https://doi.org/10.1177/0022487117732317>

- S61 Ling, U. L., Saibin, T. C., Labadin, J., & Aziz, N. A. (2018.). Preliminary Investigation: Teachers' Perception on Computational Thinking Concepts. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(2-9), 23-29.
- S62 Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, Vol. 41, pp. 51–61. <https://doi.org/10.1016/j.chb.2014.09.012>
- S63 Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, Vol. 80, pp. 470–477. <https://doi.org/10.1016/j.chb.2017.09.025>
- S64 Mishra, P., Cain, W., Sawaya, S., & Henriksen, D. (2013). Rethinking Technology & Creativity in the 21st Century: A Room of Their Own. *TechTrends*, 57(4), 5–9. <https://doi.org/10.1007/s11528-013-0668-7>
- S65 Monteiro, I. T., Salgado, L. C. de C., Mota, M. P., Sampaio, A. L., & de Souza, C. S. (2017). Signifying software engineering to computational thinking learners with AgentSheets and PoliFacets. *Journal of Visual Languages and Computing*, 40, 91–112. <https://doi.org/10.1016/j.jvlc.2017.01.005>
- S66 Moreno León, J., Robles, G., & Román González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia*, (46), 6.
- S67 Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, Vol. 33, pp. 61–76. <https://doi.org/10.14742/ajet.3521>

- S68 Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*.
<https://doi.org/10.1080/20004508.2019.1627844>
- S69 Ozturk, Z., Dooley, C. M. M., & Welch, M. (2018). Finding the Hook: Computer Science Education in Elementary Contexts. *Journal of Research on Technology in Education*, 50(2), 149–163. <https://doi.org/10.1080/15391523.2018.1431573>
- S70 Pinkard, N., Martin, C. K., & Erete, S. (2019). Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments*, 0(0), 1–15.
<https://doi.org/10.1080/10494820.2019.1636070>
- S71 Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Emerging Research, Practice, and Policy on Computational Thinking. *Emerging Research, Practice, and Policy on Computational Thinking*, 291–305.
<https://doi.org/10.1007/978-3-319-52691-1>
- S72 Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., ... Gluck, F. (2015). Scalable Game Design. *ACM Transactions on Computing Education*, 15(2), 1–31. <https://doi.org/10.1145/2700517>
- S73 Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 265–269. <https://doi.org/10.1145/1734263.1734357>
- S74 Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. <https://doi.org/10.1145/1592761.1592779>
- S75 Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2019). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*.
<https://doi.org/10.1080/10494820.2019.1612448>

- S76 Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, Vol. 72, pp. 678–691. <https://doi.org/10.1016/j.chb.2016.08.047>
- S77 Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior*, 80, 441–459. <https://doi.org/10.1016/j.chb.2017.09.030>
- S78 Román-gonzález, M., Pérez-gonzález, J., & Moreno-león, J. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*, 18, 47–58. <https://doi.org/10.1016/j.ijcci.2018.06.004>
- S79 Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using “scratch” in five schools. *Computers and Education*, 97, 129–141. <https://doi.org/10.1016/j.compedu.2016.03.003>
- S80 Selby, C. (2013). Computational Thinking: The Developing Definition. *ITiCSE Conference 2013*, 5–8.
- S81 Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, Vol. 18, pp. 351–380. <https://doi.org/10.1007/s10639-012-9240-x>
- S82 Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher’s perspective. *Education and Information Technologies*, 22(2), 469–495. <https://doi.org/10.1007/s10639-016-9482-0>
- S83 Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, Vol. 22, pp. 142–158. <https://doi.org/10.1016/j.edurev.2017.09.003>

- S84 Snow, E., Rutstein, D., Basu, S., Bienkowski, M., & Everson, H. T. (2019). Leveraging Evidence-Centered Design to Develop Assessments of Computational Thinking Practices. *International Journal of Testing*, Vol. 19, pp. 103–127. <https://doi.org/10.1080/15305058.2018.1543311>
- S85 Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, 28(2), 347–376. <https://doi.org/10.1007/s10798-017-9400-9>
- S86 von Wangenheim, C. G., Hauck, J. C. R., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F. (2018). CodeMaster - Automatic assessment and grading of app inventor and snap! Programs. *Informatics in Education*, 17(1), 117–150. <https://doi.org/10.15388/infedu.2018.08>
- S87 Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, Vol. 20, pp. 715–728. <https://doi.org/10.1007/s10639-015-9412-6>
- S88 Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, Vol. 25, pp. 127–147. <https://doi.org/10.1007/s10956-015-9581-5>
- S89 Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning*, Vol. 6, pp. 1–17. <https://doi.org/10.4018/IJGBL.2016010101>
- S90 Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). Werner, Linda Denner, Jill Campe, Shannon Kawamoto, Damon Chizuru. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. <https://doi.org/10.3758/BF03196322>
- S91 Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

- S92 Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725.
<https://doi.org/10.1098/rsta.2008.0118>
- S93 Witherspoon, E. B., & Schunn, C. D. (2019). Teachers' goals predict computational thinking gains in robotics. *Information and Learning Science*, Vol. 120, pp. 308–326. <https://doi.org/10.1108/ILS-05-2018-0035>
- S94 Xing, W. (2019). Large-scale path modeling of remixing to computational thinking. *Interactive Learning Environments*.
<https://doi.org/10.1080/10494820.2019.1573199>
- S95 Yadav, A., Mayfield, C., Zhou, N., Hambruch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, Vol. 14. <https://doi.org/10.1145/2576872>
- S96 Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, Vol. 60, pp. 55–62.
<https://doi.org/10.1145/2994591>
- S97 Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, Vol. 24, pp. 929–951.
<https://doi.org/10.1007/s10639-018-9801-8>
- S98 Yasar, O. (2018). Viewpoint a new perspective on computational thinking: Addressing its cognitive essence, universal value, and curricular practices. *Communications of the ACM*, 61(7), 33–39. <https://doi.org/10.1145/3214354>
- S99 Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, 141(June), 103607.
<https://doi.org/10.1016/j.compedu.2019.103607>
- S100 Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education*, 141(July), 103633.
<https://doi.org/10.1016/j.compedu.2019.103633>

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

Appendix B

Appendix B. List of selected studies (Chapter 5)

- PS1 Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, Vol. 23, pp. 1501–1514. <https://doi.org/10.1007/s10639-017-9675-1>
- PS2 Bui, L. D., Kim, Y. G., Ho, W., Thi, H., Ho, T., & Pham, N. K. (2018). Developing WebQuest 2.0 model for promoting computational thinking skill. In *International Journal of Engineering & Technology* (Vol. 7). Retrieved from <http://bit.ly/2jH9KT2>.
- PS3 Cachero C., Barra P., Melia S., Lopez O., "Impact of Programming Exposure on the Development of Computational Thinking Capabilities: An Empirical Study", 2020, "IEEE Access", "10.1109/ACCESS.2020.2987254", "https://www.scopus.com/inward/record.uri?eid=2-s2.0-85084334725&doi=10.1109%2fACCESS.2020.2987254&partnerID=40&md5=1333d67d45be18c0a20cbc955aa580d6"
- PS4 Cetin, I. (2016). Preservice Teachers' Introduction to Computing: Exploring Utilization of Scratch. *Journal of Educational Computing Research*, Vol. 54, pp. 997–1021. <https://doi.org/10.1177/0735633116642774>
- PS5 Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers and Education*, 95, 202–215. <https://doi.org/10.1016/j.compedu.2016.01.010>
- PS6 Choi, S.-Y. (2019). Development of an instructional model based on constructivism for fostering computational thinking. *International Journal of Innovative Technology and Exploring Engineering*, Vol. 8, pp. 381–385. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85064633190&partnerID=40&md5=9c6da0548c789dfbb24134edc2cbfdc7>

- PS7 Cutumisu, M., & Guo, Q. (2019). Using Topic Modeling to Extract Pre-Service Teachers' Understandings of Computational Thinking From Their Coding Reflections. *IEEE Transactions on Education*.
<https://doi.org/10.1109/TE.2019.2925253>
- PS8 Dolgopolas, V., & Jevsikova, T. (2015). On Evaluation of computational thinking of software engineering novice students. *Proceedings of The*, 4(2), 105–112. <https://doi.org/10.13140/RG.2.1.2855.9206>
- PS9 Fang, A.-D., Chen, G.-L., Cai, Z.-R., Cui, L., & Harn, L. (2017). Research on blending learning flipped class model in colleges and universities based on computational thinking - “Database principles” for example. *Eurasia Journal of Mathematics, Science and Technology Education*, Vol. 13, pp. 5747–5755.
<https://doi.org/10.12973/eurasia.2017.01024a>
- PS10 Fernández, J. M., Zúñiga, M. E., Rosas, M. V., & Guerrero, R. A. (2018). Experiences in Learning Problem-Solving through Computational Thinking. *Journal of Computer Science and Technology*, 18(02), e15.
<https://doi.org/10.24215/16666038.18.e15>
- PS11 Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cárdenas, L., Pantano, P., & Bilotta, E. (2019). Lesson planning by computational thinking skills in Italian pre-service teachers. *Informatics in Education*, Vol. 18, pp. 69–104.
<https://doi.org/10.15388/infedu.2019.04>
- PS12 Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *SIGCSE Bulletin Inroads*, Vol. 41, pp. 183–187.
<https://doi.org/10.1145/1539024.1508931>
- PS13 Hou H.-Y., Agrawal S., Lee C.-F., "Computational thinking training with technology for non-information undergraduates", 2020, "Thinking Skills and Creativity", "10.1016/j.tsc.2020.100720", "https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85090228702&doi=10.1016%2fj.tsc.2020.100720&partnerID=40&md5=a6c007fa4e1130f69a15118167520743"

- PS14 Huang, X.-P., & Leng, J. (2019). Design of database teaching model based on computational thinking training. *International Journal of Emerging Technologies in Learning*, Vol. 14, pp. 52–69. <https://doi.org/10.3991/ijet.v14i08.10495>
- PS15 Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology*, Vol. 26, pp. 175–192. <https://doi.org/10.1007/s10956-016-9663-z>
- PS16 Jeon, Y., & Kim, T. (2017). The effects of the computational thinking-based programming class on the computer learning attitude of non-major students in the teacher training college. *Journal of Theoretical and Applied Information Technology*, 95(17), 4330–4339.
- PS17 Kang Y., Lee K., "Designing technology entrepreneurship education using computational thinking", 2020, "Education and Information Technologies", "10.1007/s10639-020-10231-2", "https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085371936&doi=10.1007%2fs10639-020-10231-2&partnerID=40&md5=72af666d82419c9dc4344f0144fb7e18"
- PS18 Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, 47, 1991–1999. <https://doi.org/10.1016/j.sbspro.2012.06.938>
- PS19 Katai Z., "Promoting computational thinking of both sciences- and humanities-oriented students: an instructional and motivational design perspective", 2020, "Educational Technology Research and Development", "10.1007/s11423-020-09766-5", "https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85083359163&doi=10.1007%2fs11423-020-09766-5&partnerID=40&md5=d27baf0027852e2a276dfd1c052f9fe7"

- PS20 Kılıç S., Gökoğlu S., Öztürk M., "A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking", 2020, "Journal of Educational Computing Research", "10.1177/0735633120964402", "<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85092671278&doi=10.1177%2f0735633120964402&partnerID=40&md5=a767d80ae448790747589c61c0fe8ac6>"
- PS21 Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). In *Computers in Human Behavior* (Vol. 72, pp. 558–569). <https://doi.org/10.1016/j.chb.2017.01.005>
- PS22 Kwon, J., & Kim, J. (2018). A study on the design and effect of computational thinking and software education. *KSII Transactions on Internet and Information Systems*, Vol. 12, pp. 4057–4071. <https://doi.org/10.3837/tiis.2018.08.028>
- PS23 Lee, Y., & Cho, J. (2019). Knowledge representation for computational thinking using knowledge discovery computing. *Information Technology and Management*. <https://doi.org/10.1007/s10799-019-00299-9>
- PS24 Li, M., & Hou, D. (2014). Network autonomous learning based on computational thinking. *World Transactions on Engineering and Technology Education*, Vol. 12, pp. 576–580. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84916237711&partnerID=40&md5=3d31af2cb32278ebb421ef9de6f7271f>
- PS25 Lin, P. H., & Chen, S. Y. (2020). Design and Evaluation of a Deep Learning Recommendation Based Augmented Reality System for Teaching Programming and Computational Thinking. *IEEE Access*, 8, 45689–45699. <https://doi.org/10.1109/ACCESS.2020.2977679>
- PS26 Ma, J. Bin, Teng, G. F., Zhou, G. H., & Sun, C. X. (2017). Practical teaching reform on computational thinking training for undergraduates of computer major.

Eurasia Journal of Mathematics, Science and Technology Education, 13(10), 7121–7130. <https://doi.org/10.12973/ejmste/78738>

- PS27 Magana, A. J., & Silva Coutinho, G. (2017). Modeling and simulation practices for a computational thinking-enabled engineering workforce. *Computer Applications in Engineering Education*, 25(1), 62–78. <https://doi.org/10.1002/cae.21779>
- PS28 Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, Vol. 33, pp. 61–76. <https://doi.org/10.14742/ajet.3521>
- PS29 Page, R., & Gamboa, R. (2013). How Computers Work: Computational Thinking for Everyone. *Electronic Proceedings in Theoretical Computer Science*, 106(Tfpie 2012), 1–19. <https://doi.org/10.4204/eptcs.106.1>
- PS30 Pala, F. K., & Mihçı Türker, P. (2019). The effects of different programming trainings on the computational thinking skills. *Interactive Learning Environments*. <https://doi.org/10.1080/10494820.2019.1635495>
- PS31 Qin, H. (2009). Teaching computational thinking through bioinformatics to biology students. *SIGCSE Bulletin Inroads*, Vol. 41, pp. 188–191. <https://doi.org/10.1145/1539024.1508932>
- PS32 Rodríguez-García J.D., Moreno-León J., Román-González M., Robles G., "LearningML: A tool to foster computational thinking skills through practical artificial intelligence projects", 2020, "Revista de Educacion a Distancia", "10.6018/RED.410121", "<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085603174&doi=10.6018%2fRED.410121&partnerID=40&md5=8629359de325467d947de6634fd4b859>"

- PS33 Romero M, Lepage A, Lille B. Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*. 2017 Dec;14(42): 1-15
- PS34 Rubinstein, A., & Chor, B. (2014). Computational Thinking in Life Science Education. *PLoS Computational Biology*, Vol. 10. <https://doi.org/10.1371/journal.pcbi.1003897>
- PS35 Shih, H., Jackson, J. M., Hawkins-Wilson, C. L., & Yuan, P.-C. (2015). Promoting computational thinking skills in an emergency management class with MIT app inventor. *Computers in Education Journal*, Vol. 6, pp. 82–91. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85052799029&partnerID=40&md5=df2b3b5a13cf93ab789a443cd5d7dd>
- PS36 Sondakh D.E., Osman K., Zainudin S., "A proposal for holistic assessment of computational thinking for undergraduate: Content validity", 2020, "European Journal of Educational Research", "10.12973/eu-jer.9.1.33", "<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85082097163&doi=10.12973%2feu-jer.9.1.33&partnerID=40&md5=54d1017f4af582d22b1edc9970ecd68f>"
- PS37 Taylor, N. G., Moore, J., Visser, M., & Drouillard, C. (2018). Incorporating computational thinking into library graduate course goals and objectives. *School Library Research*, Vol. 21. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85053260681&partnerID=40&md5=15815233f96a5912185346fe719f6496>
- PS38 Wu, B., Hu, Y., Ruis, A. R., & Wang, M. (2019). Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning*, Vol. 35, pp. 421–434. <https://doi.org/10.1111/jcal.12348>
- PS39 Yuen, T. T., & Robbins, K. A. (2014). A qualitative study of students' computational thinking skills in a data-driven computing class. *ACM Transactions on Computing Education*, Vol. 14. <https://doi.org/10.1145/2676660>

- PS40 Zha, S., Jin, Y., Moore, Gaston J. (2020). Hopscotch into Coding: Introducing Pre-Service Teachers Computational Thinking. *TechTrends*, 64, 17–28.
<https://doi.org/10.1007/s11528-019-00423-0>
- PS41 Zha S., Jin Y., Moore P., Gaston J. (2020). A cross-institutional investigation of a flipped module on preservice teachers' interest in teaching computational thinking (2020), *Journal of Digital Learning in Teacher Education*.
<https://doi.org/10.1080/21532974.2019.1693941>

Appendix C

Appendix C. Research instruments.

Instrument adapted from Park (2009).

Δημογραφικά Στοιχεία

1. Συμπληρώσετε το σχολικό σας email *
-

2. Τάξη *

Να επισημαίνεται μόνο μία έλλειψη.

- ΓυμνασίουΑ
- ΓυμνασίουΒ
- ΓυμνασίουΓ

3. Φύλλο *

Να επισημαίνεται μόνο μία έλλειψη.

- Αγόρι
- Κορίτσι

4. Προηγούμενη εμπειρία προγραμματισμού *

Να επισημαίνεται μόνο μία έλλειψη.

- Ναι
- Όχι

Ευκολία Χρήσης

1. Βρήκα το παιχνίδι προγραμματισμού και υπολογιστικής σκέψης aMazeD εύκολο στη χρήση. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

2. Το να μάθω να χρησιμοποιώ ένα παιχνίδι προγραμματισμού και υπολογιστικής σκέψης είναι εύκολο για εμένα. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

Χρησιμότητα

1. Το παιχνίδι προγραμματισμού και υπολογιστικής σκέψης aMazeD μπορεί να με βοηθήσει να καταλάβω τις έννοιες και τις πρακτικές προγραμματισμού και υπολογιστικής σκέψης. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

2. Το παιχνίδι προγραμματισμού και υπολογιστικής σκέψης aMazeD μπορεί να κάνει ευκολότερη την μελέτη των εννοιών και των πρακτικών προγραμματισμού και υπολογιστικής σκέψης.*

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

3. Οι συμβουλές που μου παρείχε το παιχνίδι ήταν επαρκείς για να με βοηθήσουν να επιλύσω τα επίπεδα.*

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

4. Οι συμβουλές που μου παρείχε το παιχνίδι ήταν χρήσιμες για να με βοηθήσουν να επιλύσω τα επίπεδα. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

5. Οι συμβουλές που μου παρείχε το παιχνίδι με βοήθησαν να κατανοήσω τις βασικές έννοιες προγραμματισμού και υπολογιστικής σκέψης. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

Στάση

1. Η μελέτη προγραμματισμού και υπολογιστικής σκέψης μέσω παιχνιδιών όπως το aMazeD είναι μια καλή ιδέα. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

2. Είμαι θετικός/θετική απέναντι στα παιχνίδια προγραμματισμού και υπολογιστικής σκέψης. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

Προσιτότητα

Δεν αντιμετώπισα καμία δυσκολία στη χρήση του παιχνιδιού προγραμματισμού. *

Να επισημαίνεται μόνο μία έλλειψη.

Διαφωνώ έντονα

1

2

3

4

5

Συμφωνώ έντονα

Συνολική Εμπειρία

Γράψτε λίγα λόγια για την εμπειρία σας από το παιχνίδι. Τί σας άρεσε ή δεν σας άρεσε; Τί σας έκανε εντύπωση;*

Scale of Attitudes towards Programming adapted from Kong (2018)

1. Συμπληρώσετε το σχολικό σας email *

2. Τάξη *

Να επισημαίνεται μόνο μία έλλειψη.

ΓυμνασίουΑ

ΓυμνασίουΒ

ΓυμνασίουΓ

3. Φύλλο *

Να επισημαίνεται μόνο μία έλλειψη.

Αγόρι

Κορίτσι

Meaningfulness

1. Ο προγραμματισμός είναι χρήσιμος για εμένα. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

1

2

3

4

5

Διαφωνώ απόλυτα

2. Ο προγραμματισμός θα με βοηθήσει να πετύχω τους στόχους μου. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

3. Θέλω να γίνω καλός στον προγραμματισμό. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

4. Ο προγραμματισμός είναι σημαντικός για μένα. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

Programming self-efficacy

1. Μπορώ να μάθω πώς να προγραμματίζω. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

2. Είμαι καλός στον προγραμματισμό. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

3. Σκέφτομαι τον εαυτό μου ως κάποιον που μπορεί να προγραμματίσει. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

4. Έχω δεξιότητες προγραμματισμού. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

5. Έχω εμπιστοσύνη στην ικανότητά μου να προγραμματίζω. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4
- 5

Διαφωνώ απόλυτα

Interest in programming

1. Ο προγραμματισμός είναι ενδιαφέρων. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

- 1
- 2
- 3
- 4

5

Διαφωνώ απόλυτα

2. Είμαι περίεργος για το περιεχόμενο του προγραμματισμού. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

1

2

3

4

5

Διαφωνώ απόλυτα

3. Νομίζω ότι ο προγραμματισμός είναι διασκεδαστικός. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

1

2

3

4

5

Διαφωνώ απόλυτα

4. Με ενδιαφέρουν πολύ οι δραστηριότητες προγραμματισμού υπολογιστών. *

Να επισημαίνεται μόνο μία έλλειψη.

Συμφωνώ απόλυτα

1

2

3

4

5

Διαφωνώ απόλυτα

Computational Thinking Test (CTt) adopted from Román-González et al. (2017)

Καλώς ήρθατε στη δοκιμασία Υπολογιστικής Σκέψης!

*** Υποδεικνύει απαιτούμενη ερώτηση**

1. Διεύθυνση ηλεκτρονικού ταχυδρομείου *

2. Φύλλο *

Να επισημαίνεται μόνο μία έλλειψη.

Αγόρι

Κορίτσι

3. Τάξη *

Να επισημαίνεται μόνο μία έλλειψη.

ΓυμνασίουΑ

ΓυμνασίουΒ

ΓυμνασίουΓ

Οδηγίες

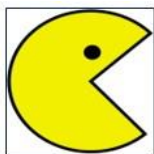
Η δοκιμασία αποτελείται από 28 ερωτήσεις, σε 7 σελίδες με 4 ερωτήσεις η καθεμία.

Όλες οι ερωτήσεις έχουν 4 επιλογές απαντήσεων (Α, Β, C ή D) από τις οποίες μόνο μία είναι σωστή.

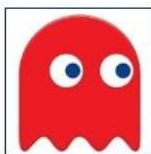
Έχετε 45 λεπτά να κάνετε το καλύτερο που μπορείτε. Δεν είναι απαραίτητο να απαντήσετε σε όλες τις ερωτήσεις.

Για να προχωρήσετε σε επόμενη σελίδα επιλέξτε "Συνέχεια" στο κάτω μέρος της σελίδας. ΠΟΛΥ ΣΗΜΑΝΤΙΚΟ: όταν ολοκληρώσετε τη δοκιμασία ή ο χρόνος σας τελειώσει θα πρέπει να μετακινηθείτε στην τελευταία σελίδα και να επιλέξετε "Υποβολή".

Πριν ξεκινήσετε τη δοκιμασία μπορείτε να δείτε παρακάτω παραδείγματα ερωτήσεων σαν αυτά που θα πρέπει να απαντήσετε.



‘Pac-Man’



Ghost



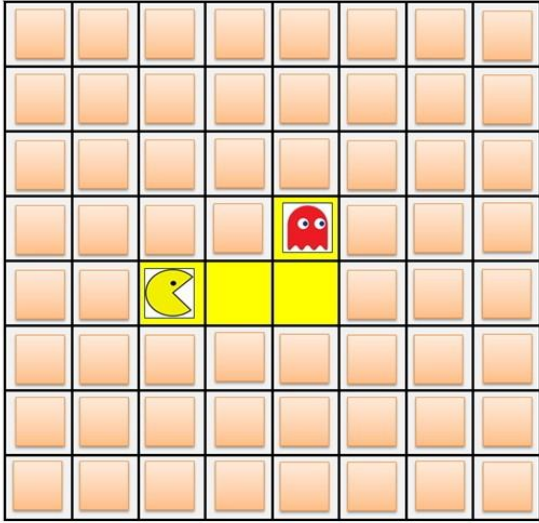





Artist

Παράδειγμα 1

Σε αυτό το παράδειγμα σας ζητείται να δώσετε οδηγίες έτσι ώστε το Pac-man να συναντήσει το φάντασμα (ghost) μέσω του κίτρινου μονοπατιού.

Η σωστή απάντηση είναι το B.

Παράδειγμα 1. Ποιες οδηγίες πρέπει να δοθούν στο Pac-man έτσι ώστε να μεταβεί στο φάντασμα (ghost) μέσω του κίτρινου μονοπατιού;

<p>Which instructions take 'Pac-Man' to the ghost by the path marked out?</p> 	<p>Option A</p>  <hr/> <p>Option B</p>   <hr/> <p>Option C</p>  <hr/> <p>Option D</p> 
--	---

Παράδειγμα 1 *

Επιλέξτε την απάντηση B

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Παράδειγμα 2.

Και σε αυτό το παράδειγμα σας ζητείται να δώσετε οδηγίες έτσι ώστε το Pac-man να μεταβεί στο φάντασμα (ghost) μέσω του κίτρινου μονοπατιού. Εδώ όμως οι απαντήσεις παρουσιάζονται ως οδηγίες και όχι ως βέλη.

Η σωστή απάντηση είναι το C.

Παράδειγμα 2. Ποιες οδηγίες πρέπει να δοθούν στο Pac-man έτσι ώστε να μεταβεί στο φάντασμα (ghost) μέσω του κίτρινου μονοπατιού;

Which instructions take 'Pac-Man' to the ghost by the path marked out?

	<p>Option A</p> <p>όταν εκτελείται</p> <p>κινήσου μπροστά</p> <p>στρίψε αριστερά</p> <p>κινήσου μπροστά</p> <p>κινήσου μπροστά</p>	<p>Option B</p> <p>όταν εκτελείται</p> <p>κινήσου μπροστά</p> <p>στρίψε δεξιά</p> <p>κινήσου μπροστά</p> <p>κινήσου μπροστά</p>
	<p>Option C</p> <p>όταν εκτελείται</p> <p>κινήσου μπροστά</p> <p>κινήσου μπροστά</p> <p>στρίψε αριστερά</p> <p>κινήσου μπροστά</p>	<p>Option D</p> <p>όταν εκτελείται</p> <p>κινήσου μπροστά</p> <p>κινήσου μπροστά</p> <p>στρίψε δεξιά</p> <p>κινήσου μπροστά</p>

A green checkmark is placed in the center of the bottom row of the options table, indicating that Option C is the correct answer.

Παράδειγμα 2 *

Επιλέξτε την απάντηση C.

Να επισημαίνεται μόνο μία έλλειψη.

A

B

C

D


Παράδειγμα 3

Στο Παράδειγμα 3 σας ζητείται να δώσετε οδηγίες έτσι ώστε ο καλλιτέχνης (artist) να σχεδιάσει το σχήμα της οθόνης.

Με την εντολή ΚΙΝΗΣΟΥ ο καλλιτέχνης (artist) προχωρά και ζωγραφίζει ενώ με την εντολή ΜΕΤΑΒΑΣΗ ο καλλιτέχνης (artist) πηδά χωρίς να ζωγραφίζει.

Το γκρι βέλος δείχνει την κατεύθυνση εκκίνησης του καλλιτέχνη. Η σωστή απάντηση είναι το Α.

Παράδειγμα 3. Ποιες οδηγίες πρέπει να δοθούν στον καλλιτέχνη έτσι ώστε να σχεδιάσει το παρακάτω σχήμα; Η μικρή πλευρά είναι 50 εικονοστοιχεία και η μεγάλη 100.

<p>Which instructions should the artist follow to draw the shape? The short side measures 50 pixels and the long side measures 100 pixels.</p> 	<p>Option A</p> <p>όταν εκτελεστεί</p> <p>κινήσου μπροστά κατά 50 εικονοστοιχεία</p> <p>στρίψε αριστερά κατά 90 μοίρες</p> <p>κινήσου μπροστά κατά 100 εικονοστοιχεία</p>	<p>Option B</p> <p>όταν εκτελεστεί</p> <p>κινήσου μπροστά κατά 50 εικονοστοιχεία</p> <p>στρίψε αριστερά κατά 90 μοίρες</p> <p>κινήσου μπροστά κατά 100 εικονοστοιχεία</p>
	<p>Option C</p> <p>όταν εκτελεστεί</p> <p>κινήσου μπροστά κατά 50 εικονοστοιχεία</p> <p>στρίψε δεξιά κατά 90 μοίρες</p> <p>κινήσου μπροστά κατά 100 εικονοστοιχεία</p>	<p>Option D</p> <p>όταν εκτελεστεί</p> <p>κινήσου μπροστά κατά 100 εικονοστοιχεία</p> <p>στρίψε δεξιά κατά 90 μοίρες</p> <p>κινήσου μπροστά κατά 50 εικονοστοιχεία</p>

Παράδειγμα 3 *

Επιλέξτε την απάντηση Α.

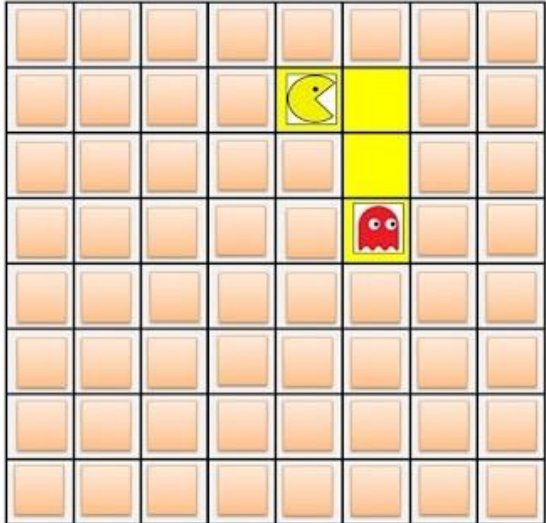
Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 1-4

Ερώτηση 1. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

Which instructions take 'Pac-Man' to the ghost by the path marked out?



Option A
→ → ↓

Option B
→ ↓ ↓

Option C
→ → ↓ ↓

Option D
↓ ↓ →

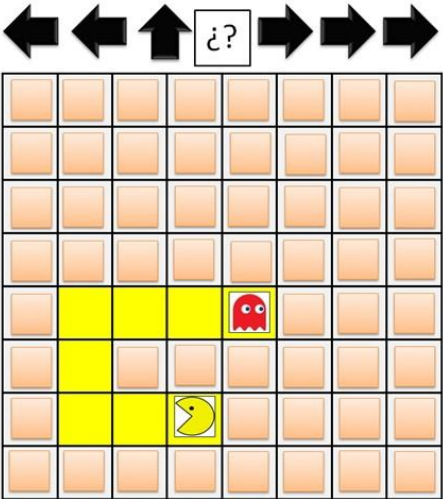




Ερώτηση 1

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 2. Ποιο βήμα λείπει στις παρακάτω οδηγίες για να μεταβεί το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

<p>Which step is missing in the instructions below to take 'Pac-Man' to the ghost by the path marked out?</p> 	<p>Option A</p> 
	<p>Option B</p> 
	<p>Option C</p> 
	<p>Option D</p> 

Ερώτηση 2

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 3. Οι οδηγίες πρέπει να οδηγήσουν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι. Σε ποιο βήμα των οδηγιών υπάρχει κάποιο λάθος;

The instructions should take 'Pac-Man' to the ghost by the path marked out. In which step of the instructions is there a **mistake**?

Ερώτηση 3

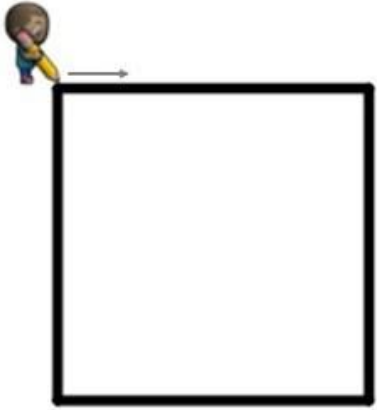
Επιλέξτε το βήμα στο οποίο υπάρχει λάθος

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 4. Ποιες οδηγίες πρέπει να ακολουθήσει ο καλλιτέχνης για να σχεδιάσει την πλατεία; Κάθε μία από τις πλευρές του τετραγώνου έχει μέγεθος 100 εικονοστοιχεία.

Which instructions should the artist follow to draw the square? Each of the sides of the square measures 100 pixels.



Option A

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε αριστερά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

Option B

κινήσου μπροστά κατά 25 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 25 εικονοστοιχεία

στρίψε αριστερά κατά 90 μοίρες

κινήσου μπροστά κατά 25 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 25 εικονοστοιχεία

Option C

κινήσου μπροστά κατά 50 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 50 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 50 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 50 εικονοστοιχεία

Option D

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

στρίψε δεξιά κατά 90 μοίρες

κινήσου μπροστά κατά 100 εικονοστοιχεία

Ερώτηση 4

Επιλέξτε τη σωστή απάντηση

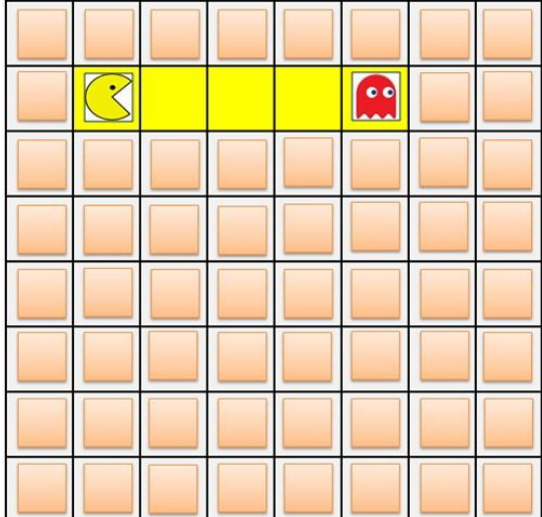

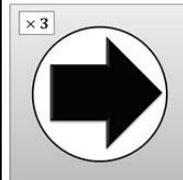


Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 5-8

Ερώτηση 5. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

Which instructions take 'Pac-Man' to the ghost by the path marked out?

	<p>Option A</p> 	<p>Option B</p> 
	<p>Option C</p> 	<p>Option D</p> 

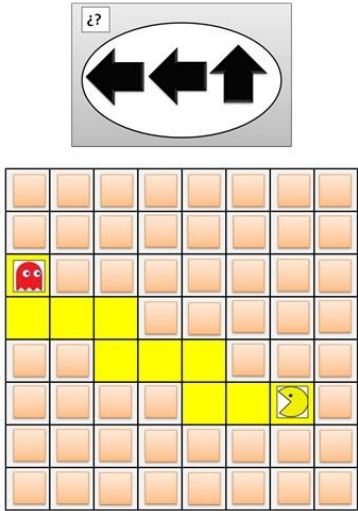
Ερώτηση 5

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 6. Πόσες φορές πρέπει να επαναληφθεί η ακολουθία για να οδηγηθεί το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

<p><i>How many times must the sequence be repeated to take 'Pac-Man' to the ghost by the path marked out?</i></p> 	<p>Option A × 2</p>
	<p>Option B × 1</p>
	<p>Option C × 4</p>
	<p>Option D × 3</p>

Ερώτηση 6


Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

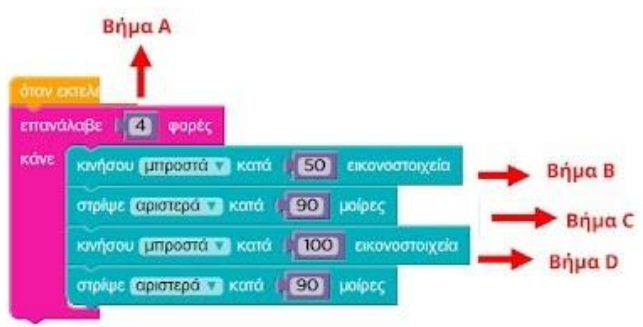
- A
- B
- C
- D

Ερώτηση 7. Οι οδηγίες πρέπει να κάνουν τον καλλιτέχνη να σχεδιάσει το ακόλουθο ορθογώνιο μία φορά (πλάτος 50 εικονοστοιχεία και ύψος 100 εικονοστοιχεία). Σε ποιο βήμα των οδηγιών υπάρχει κάποιο λάθος;

*The instructions should make the artist draw the following rectangle **once** (50 pixels wide and 100 pixels high). In which step of the instructions is there a **mistake**?*



Βήμα A



Ερώτηση 7

Επιλέξτε το βήμα στο οποίο υπάρχει λάθος

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 8. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

Which instructions take 'Pac-Man' to the ghost by the path marked out?

Option A

```

όταν εκτελεστεί
  επανάλαβε 4 φορές
    κάνε επανάλαβε 3 φορές
      κινήσου μπροστά
    στρίψε δεξιά
  κινήσου μπροστά
        
```

Option B

```

όταν εκτελεστεί
  επανάλαβε 3 φορές
    κάνε επανάλαβε 4 φορές
      κινήσου μπροστά
    στρίψε δεξιά
  κινήσου μπροστά
        
```

Option C

```

όταν εκτελεστεί
  επανάλαβε 3 φορές
    κάνε επανάλαβε 4 φορές
      κινήσου μπροστά
    στρίψε δεξιά
  κινήσου μπροστά
        
```

Option D

```

όταν εκτελεστεί
  επανάλαβε 4 φορές
    κάνε προχώρησε μπροστά
  επανάλαβε 3 φορές
    κάνε στρίψε δεξιά
  προχώρησε μπροστά
        
```

Ερώτηση 8

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 9-12

Ερώτηση 9. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

Which instructions take 'Pac-Man' to the ghost by the path marked out?

Option A
Επανάλαβε Μέχρι.....

Option B
Επανάλαβε Μέχρι.....

Option C
Επανάλαβε Μέχρι.....

Option D
Επανάλαβε Μέχρι.....

Ερώτηση 9

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 10. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

	<p>Option A</p>	<p>Option B</p>
	<p>Option C</p>	<p>Option D</p> <p>Δεν λείπει κάποιο βήμα.</p>

Ερώτηση 10

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 11. Οι οδηγίες πρέπει να οδηγήσουν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι. Σε ποιο βήμα των οδηγιών υπάρχει κάποιο λάθος;

The instructions should take 'Pac-Man' to the ghost by the path marked out. In which step of the instructions is there a **mistake**?

Βήμα Α
Βήμα Β
Βήμα C
Βήμα D

Επανάλαβε Μέχρι...
x 2


Ερώτηση 11

Επιλέξτε το βήμα στο οποίο υπάρχει λάθος

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 12. Ποιες οδηγίες πρέπει να δοθούν στον καλλιτέχνη για να σχεδιάσει τη σκάλα που φαίνεται στην παρακάτω εικόνα; Ανάμεσα σε κάθε σκαλί μεσολαβεί κενό 30 εικονοστοιχεία.

<p>Which instructions should the artist follow to draw the ladder that reaches the flower? There are 30 pixels between each rung.</p> 	<p>Option A</p> <pre> όταν εκτελεστεί επανάλαβε μέχρι το λουλούδι κάνε επανάλαβε 4 φορές κάνε κινήσου μπροστά κατά 30 εικονοστοιχεία στρίψε δεξιά κατά 90 μοίρες μετάβαση μπροστά κατά 30 pixels </pre>	<p>Option B</p> <pre> όταν εκτελεστεί επανάλαβε μέχρι το λουλούδι κάνε επανάλαβε 4 φορές κάνε κινήσου μπροστά κατά 120 εικονοστοιχεία στρίψε δεξιά κατά 90 μοίρες μετάβαση μπροστά κατά 30 pixels </pre>
	<p>Option C</p> <pre> όταν εκτελεστεί επανάλαβε μέχρι το λουλούδι κάνε επανάλαβε 4 φορές κάνε κινήσου μπροστά κατά 120 εικονοστοιχεία στρίψε δεξιά κατά 90 μοίρες μετάβαση μπροστά κατά 30 pixels </pre>	<p>Option D</p> <pre> όταν εκτελεστεί επανάλαβε μέχρι το λουλούδι κάνε επανάλαβε 4 φορές κάνε κινήσου μπροστά κατά 30 εικονοστοιχεία στρίψε δεξιά κατά 90 μοίρες μετάβαση μπροστά κατά 210 pixels </pre>

Ερώτηση 12

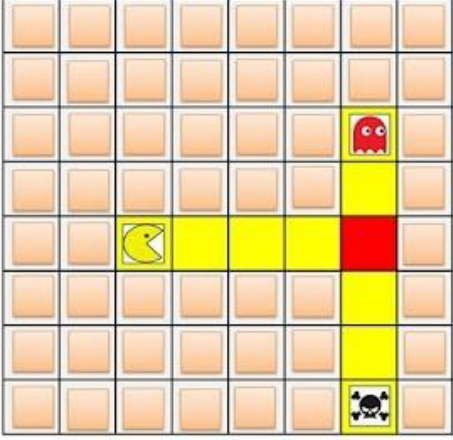
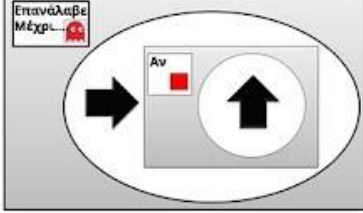
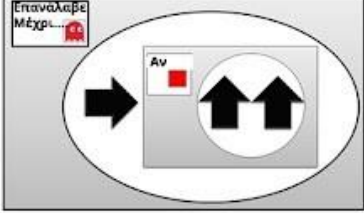
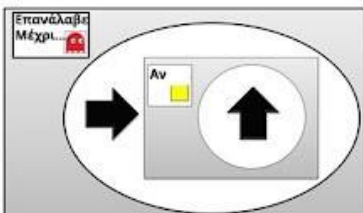
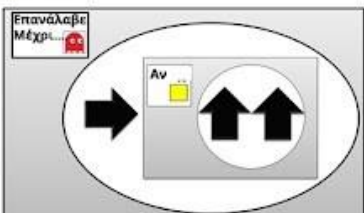
Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 13-16

Ερώτηση 13. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

<p>Which instructions take 'Pac-Man' to the ghost by the path marked out?</p> 	<p>Option A</p> 	<p>Option B</p> 
<p>Option C</p> 	<p>Option D</p> 	

Ερώτηση 13

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 15. Ποιες οδηγίες λείπουν έτσι ώστε το Pac-Man να μεταβεί στο φάντασμα από το κίτρινο μονοπάτι;

What is missing in the instructions below to take 'Pac-Man' to the ghost by the path marked out?

Option A

Option B

Option C

Option D
Και το A και το C είναι σωστά.

Ερώτηση 15

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 16. Οι οδηγίες πρέπει να οδηγήσουν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι. Σε ποιο βήμα των οδηγιών υπάρχει κάποιο λάθος;

The instructions should take 'Pac-Man' to the ghost by the path marked out.
In which step of the instructions is there a **mistake**?

Ερώτηση 16

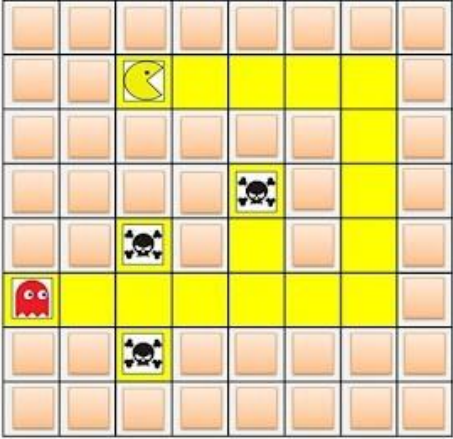










Επιλέξτε το βήμα στο οποίο υπάρχει λάθος

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 17-20

Ερώτηση 17. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

<p>Which instructions take 'Pac-Man' to the ghost by the path marked out?</p> 	<p>Option A</p> <p>όταν εκτελεστεί</p> <p>επανάλαβε μέχρι </p> <p>κάνε</p> <ul style="list-style-type: none">Εάν υπάρχει διαδρομή μπροστάκάνε προχώρησε μπροστάαλλιώς στρίψε αριστερά 	<p>Option B</p> <p>όταν εκτελεστεί</p> <p>επανάλαβε μέχρι </p> <p>κάνε</p> <ul style="list-style-type: none">Εάν υπάρχει διαδρομή μπροστάκάνε προχώρησε μπροστάαλλιώς στρίψε δεξιά 
	<p>Option C</p> <p>όταν εκτελεστεί</p> <p>επανάλαβε μέχρι </p> <p>κάνε</p> <ul style="list-style-type: none">Εάν υπάρχει μονοπάτι δεξιά κάνε στρίψε δεξιά αλλιώς προχώρησε μπροστά	<p>Option D</p> <p>όταν εκτελεστεί</p> <p>επανάλαβε μέχρι </p> <p>κάνε</p> <ul style="list-style-type: none">Εάν υπάρχει μονοπάτι αριστερά κάνε στρίψε αριστερά αλλιώς προχώρησε μπροστά

Ερώτηση 17

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 18. Ποιες οδηγίες οδηγούν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι;

<p>Which instructions take Pac-Man to the ghost by the path marked out?</p>	<p>Option A</p>	<p>Option B</p>
	<p>Option C</p>	<p>Option D</p>

Ερώτηση 18

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 19. Οι οδηγίες πρέπει να οδηγήσουν το Pac-Man στο φάντασμα από το κίτρινο μονοπάτι. Σε ποιο βήμα των οδηγιών υπάρχει κάποιο λάθος;

The instructions should take 'Pac-Man' to the ghost by the path marked out. In which step of the instructions is there a *mistake*?

Ερώτηση 19

Επιλέξτε το βήμα στο οποίο υπάρχει λάθος

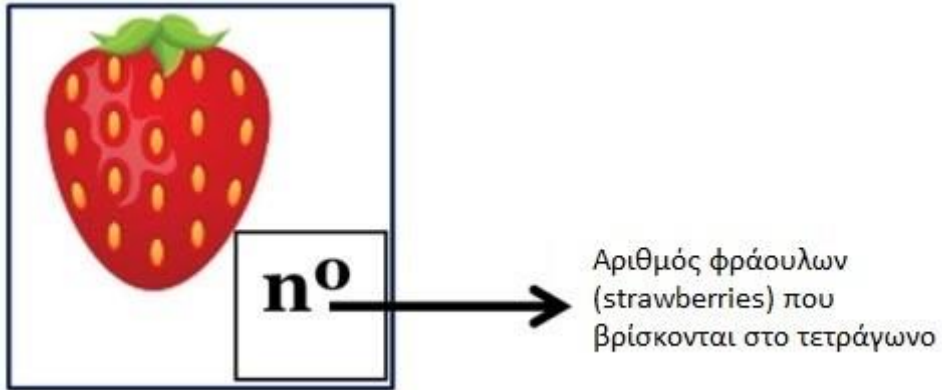
Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 21-21

ΣΗΜΑΝΤΙΚΟ: ΔΙΑΒΑΣΤΕ ΠΡΟΣΕΚΤΙΚΑ

Στις παρακάτω ερωτήσεις εμφανίζεται η εικόνα μίας φράουλας μέσα σε ένα τετράγωνο. Το τετράγωνο κάτω δεξιά εμφανίζει πόσες φράουλες υπάρχουν στο τετράγωνο.



Ερώτηση 21. Ποιες οδηγίες οδηγούν το Pac-Man στις φράουλες από το κίτρινο μονοπάτι και του λένε να φάει όλες τις φράουλες που εμφανίζονται;

<p>Which instructions take 'Pac-Man' to the strawberries by the path marked out and tell 'Pac-Man' to eat all the strawberries shown?</p>	<p>Option A</p> <pre> όταν εκτελεστεί όσο υπάρχει μονοπάτι κάνε προχώρησε μπροστά επανάλαβε 3 φορές κάνε φάε 1 φράουλα </pre>	<p>Option B</p> <pre> όταν εκτελεστεί όσο υπάρχει μονοπάτι κάνε προχώρησε μπροστά επανάλαβε 4 φορές κάνε φάε 1 φράουλα </pre>
	<p>Option C</p> <pre> όταν εκτελεστεί όσο υπάρχει μονοπάτι κάνε προχώρησε μπροστά επανάλαβε 5 φορές κάνε φάε 1 φράουλα </pre>	<p>Option D</p> <pre> όταν εκτελεστεί όσο υπάρχει μονοπάτι κάνε προχώρησε μπροστά επανάλαβε 3 φορές κάνε φάε 1 φράουλα </pre>

Ερώτηση 21

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 22. Ποιες οδηγίες οδηγούν το Pac-Man στις φράουλες από το κίτρινο μονοπάτι και του λένε να φάει όλες τις φράουλες που εμφανίζονται;

Which instructions take 'Pac-Man' to the strawberries by the path marked out and tell 'Pac-Man' to eat all the strawberries shown?

Option A	Option B
<p>όταν ειστελείται</p> <p>όσο υπάρχει μονοπάτι</p> <p>κάνε επανάλαβε 5 φορές</p> <p>κάνε προχώρησε μπροστά</p> <p>επανάλαβε 3 φορές</p> <p>κάνε φάε 1 φράουλα</p>	<p>όταν εκτελείται</p> <p>όσο υπάρχει μονοπάτι</p> <p>κάνε προχώρησε μπροστά</p> <p>επανάλαβε 3 φορές</p> <p>κάνε φάε 1 φράουλα</p>
Option C	Option D
<p>όταν ειστελείται</p> <p>όσο υπάρχει μονοπάτι</p> <p>κάνε επανάλαβε 5 φορές</p> <p>κάνε προχώρησε μπροστά</p> <p>επανάλαβε 5 φορές</p> <p>κάνε φάε 1 φράουλα</p>	<p>όταν εκτελείται</p> <p>όσο υπάρχει μονοπάτι</p> <p>κάνε προχώρησε μπροστά</p> <p>επανάλαβε 3 φορές</p> <p>κάνε φάε 1 φράουλα</p>


Ερώτηση 22

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 23. Ποιες οδηγίες λείπουν έτσι ώστε το Pac-Man να μεταβεί στις φράουλες από το κίτρινο μονοπάτι και να φάει όλες τις φράουλες που εμφανίζονται;

<p>What is missing in the instructions below to take 'Pac-Man' to the strawberries by the path marked out and tell 'Pac-Man' to eat all the strawberries shown?</p> 	<p>Option A 1 Φορά</p>
	<p>Option B 2 φορές</p>
	<p>Option C 3 φορές</p>
	<p>Option D 5 φορές</p>

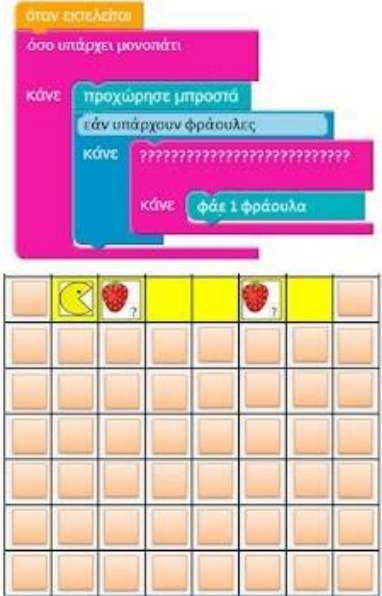
Ερώτηση 23

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 24. Ποιες οδηγίες λείπουν έτσι ώστε το Pac-Man να μεταβεί στις φράουλες από το κίτρινο μονοπάτι και να φάει όλες τις φράουλες που εμφανίζονται (αγνώστου αριθμού);

 <p>The image shows a Scratch script and a Pac-Man grid. The script starts with an orange 'when green flag clicked' block, followed by a pink 'if there is a path' block. Inside the 'if' block, there is a blue 'move forward' block, a light blue 'if there are strawberries' block, and a pink 'do' block containing a series of question marks. Below the 'if' block is a blue 'eat 1 strawberry' block. The grid below shows a 10x10 grid with a yellow path starting at (1,2) and ending at (1,7). There are strawberries at (1,3) and (1,7). The Pac-Man character is at (1,2).</p>	<p>Option A</p> <p>όσο υπάρχει μονοπάτι</p> <hr/> <p>Option B</p> <p>όσο δεν υπάρχει μονοπάτι</p> <hr/> <p>Option C</p> <p>όσο υπάρχουν φράουλες</p> <hr/> <p>Option D</p> <p>όσο δεν υπάρχουν φράουλες</p>
---	---

Ερώτηση 24

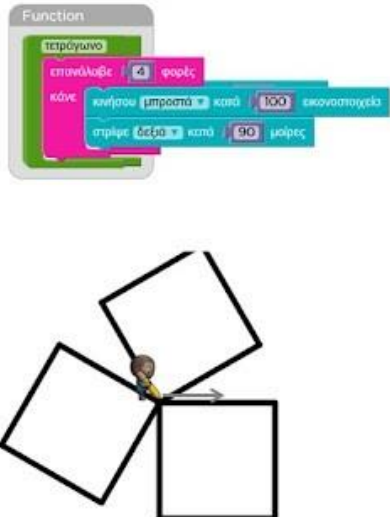




Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερωτήσεις 25 - 28

Ερώτηση 25. Ποιες οδηγίες πρέπει να ακολουθήσει ο καλλιτέχνης για να σχεδιάσει το παρακάτω σχήμα; Το σύνολο οδηγιών που εμφανίζεται στο αριστερό μέρος της εικόνας ονομάζεται συνάρτηση (function) και ζωγραφίζει ένα τετράγωνο με πλευρά 100 εικονοστοιχεία.

	<p>Option A</p> 	<p>Option B</p> 
	<p>Option C</p> 	<p>Option D</p> 

Ερώτηση 25

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 26. Οι παρακάτω οδηγίες πρέπει να κάνουν τον καλλιτέχνη να σχεδιάσει το παρακάτω σχήμα. Ποιο νούμερο λείπει στις οδηγίες; Η συνάρτηση που εμφανίζεται στο αριστερό μέρος της εικόνας ζωγραφίζει ένα τρίγωνο με πλευρά 50 εικονοστοιχεία.

<p>The following set of instructions is called 'my function', and draws one triangle of 50 pixels each side:</p> <p>The code for 'my function' is: <ul style="list-style-type: none"> Function επανάλαβε 3 φορές κάνε <ul style="list-style-type: none"> κίνησου μπροστά κατά μήκος 50 εικονοστοιχεία στρίψε αριστερά κατά 120 μοίρες </p>	<p>Option A</p> <p>15</p>	<p>Option B</p> <p>5</p>
<p>The instructions below should make the artist draw the following design. Each side of each triangle measures 50 pixels. What is missing in the instructions?</p> <p>The code is: <ul style="list-style-type: none"> όταν εκτελείται επανάλαβε ??? φορές κάνε <ul style="list-style-type: none"> τρίγωνο επεξεργασία μετάβαση μπροστά κατά 50 pixels <p>The drawing shows a horizontal line with five triangles pointing upwards, each with a side length of 50 pixels.</p> </p>	<p>Option C</p> <p>4</p>	<p>Option D</p> <p>3</p>


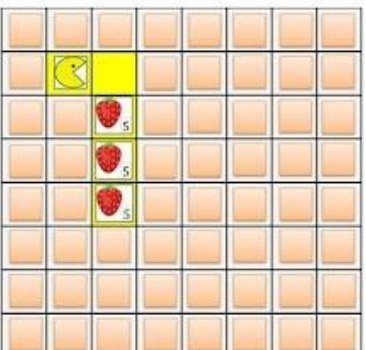
Ερώτηση 26

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 27. Ποιες οδηγίες οδηγούν το Pac-Man στις φράουλες από το κίτρινο μονοπάτι και του λένε να φάει όλες τις φράουλες που εμφανίζονται;

 	<p>Option A</p> <pre> όταν εκτελεστεί προχώρησε μπροστά στρίψε δεξιά 90 επανάλαβε 5 φορές κάνε προχώρησε μπροστά πάρει 5 </pre>	<p>Option B</p> <pre> όταν εκτελεστεί προχώρησε μπροστά στρίψε δεξιά 90 επανάλαβε 5 φορές κάνε πάρει 5 προχώρησε μπροστά </pre>
	<p>Option C</p> <pre> όταν εκτελεστεί προχώρησε μπροστά στρίψε δεξιά 90 επανάλαβε 5 φορές κάνε προχώρησε μπροστά πάρει 5 </pre>	<p>Option D</p> <pre> όταν εκτελεστεί προχώρησε μπροστά στρίψε δεξιά 90 επανάλαβε 5 φορές κάνε πάρει 5 προχώρησε μπροστά </pre>


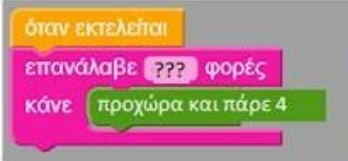
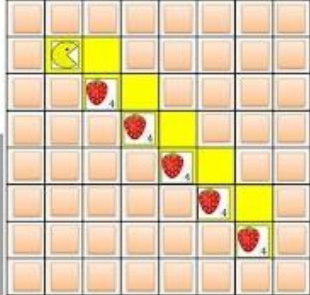
Ερώτηση 27

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

Ερώτηση 28. Ποιες οδηγίες λείπουν έτσι ώστε το Pac-Man να μεταβεί στις φράουλες από το κίτρινο μονοπάτι και να φάει όλες τις φράουλες που εμφανίζονται;

	<p>Option A</p> <p>3</p>	<p>Option B</p> <p>4</p>
 	<p>Option C</p> <p>5</p>	<p>Option D</p> <p>6</p>

Ερώτηση 28

Επιλέξτε τη σωστή απάντηση

Να επισημαίνεται μόνο μία έλλειψη.

- A
- B
- C
- D

References

- Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, 105. <https://doi.org/10.1016/j.chb.2019.03.018>
- Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' CT skills through modeling and simulations. *Education and Information Technologies*, 23, 1501–1514. <https://doi.org/10.1007/s10639-017-9675-1>
- Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, Vol. 55, pp. 832–835. <https://doi.org/10.1093/comjnl/bxs074>
- Alfayez, A. A., & Lambert, J. (2019). Exploring Saudi Computer Science Teachers' Conceptual Mastery Level of CT Skills. *Computers in the Schools*, 36, 143–166. <https://doi.org/10.1080/07380569.2019.1639593>
- Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, 19, 30–55. <https://doi.org/10.1016/j.ijcci.2018.10.004>
- Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, 105. <https://doi.org/10.1016/j.chb.2019.106185>
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 CT curriculum framework: Implications for teacher knowledge. *Educational Technology and Society*, 19, 47–57. <https://doi.org/10.1016/j.chb.2019.106185>
- Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' CT skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661–670. <https://doi.org/10.1016/j.robot.2015.10.008>

- Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How Does the Degree of Guidance Support Students' Metacognitive and Problem Solving Skills in Educational Robotics? *Journal of Science Education and Technology*, 27(1), 70–85.
<https://doi.org/10.1007/s10956-017-9709-x>
- Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *Proceedings - Frontiers in Education Conference, FIE*.
<https://doi.org/10.1109/FIE.2012.6462365>
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54. <https://doi.org/10.1145/1929887.1929905>
- Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior*, 80, 412–419. <https://doi.org/10.1016/j.chb.2017.04.058>
- Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a Computational Thinking-based science learning environment. *User Modeling and User-Adapted Interaction*, Vol. 27, pp. 5–53. <https://doi.org/10.1007/s11257-017-9187-0>
- Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the CT pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, 42(3), 53–72. <https://doi.org/10.14221/ajte.2017v42n3.4>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting*, Vancouver, BC, Canada, 1–25.
http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

- Bui, L. D., Kim, Y. G., Ho, W., Ho, H. T. T., & Pham, N. K. (2018). Developing WebQuest 2.0 model for promoting CT skill. *International Journal of Engineering and Technology (UAE)*, 7(2), 140–144. <https://doi.org/10.14419/ijet.v7i2.29.13304>
- Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, Vol. 87, pp. 834–860. <https://doi.org/10.3102/0034654317710096>
- Cachero, C., Barra, P., Melia, S., & Lopez, O. (2020). Impact of Programming Exposure on the Development of Computational Thinking Capabilities: An Empirical Study. *IEEE ACCESS*, 8, 72316–72325. <https://doi.org/10.1109/ACCESS.2020.2987254>
- Carlborg, N., Tyrén, M., Heath, C., & Eriksson, E. (2019). The scope of autonomy when teaching computational thinking in primary school. *International Journal of Child-Computer Interaction*, 21, 130–139. <https://doi.org/10.1016/j.ijcci.2019.06.005>
- Cetin, I., & Ozden, M. (2015). Development of computer programming attitude scale for university students. *Computer Applications in Engineering Education*, 23, 667–672. <https://doi.org/10.1002/cae.21639>
- Cetin, I. (2016). Preservice Teachers' Introduction to Computing: Exploring Utilization of Scratch. *Journal of Educational Computing Research*, 54, 997–1021. <https://doi.org/10.1177/0735633116642774>
- Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, 17, 93–100. <https://doi.org/10.1016/j.ijcci.2018.06.005>
- Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers and Education*, 95, 202–215. <https://doi.org/10.1016/j.compedu.2016.01.010>

- Chen, C. H., Liu, T. K. and Huang, K. (2021). ‘Scaffolding vocational high school students’ computational thinking with cognitive and metacognitive prompts in learning about programmable logic controllers’, *Journal of Research on Technology in Education*, 0(0), pp. 1–18. <https://doi.org/10.1080/15391523.2021.1983894>
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students’ computational thinking in everyday reasoning and robotics programming. *Computers and Education*, Vol. 109, pp. 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- Chevalier, M., et al. (2022). The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. *Computers and Education*, 180, 104431. <https://www.sciencedirect.com/science/article/pii/S0360131522000021>
- Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, Vol. 62, pp. 563–573. <https://doi.org/10.1007/s11528-018-0292-7>
- Choi, S.-Y. (2019). Development of an instructional model based on constructivism for fostering CT. *International Journal of Innovative Technology and Exploring Engineering*, 8, 381–385. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85064633190&partnerID=40&md5=9c6da0548c789dfbb24134edc2cbfdc7>
- Clark, D. B., & Sengupta, P. (2019). Reconceptualizing games for integrating computational thinking and science as practice: collaborative agent-based disciplinarily-integrated games. In *Interactive Learning Environments*. <https://doi.org/10.1080/10494820.2019.1636071>

- Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). Education: A future for computing education research. *Communications of the ACM*, 57(11), 34–36.
<https://doi.org/10.1145/2668899>
- Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. Retrieved from Computing at Schools. website: <https://community.computingatschool.org.uk/resources/2324/single>
- Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education*, Vol. 18, pp. 41–67. <https://doi.org/10.15388/infedu.2019.03>
- CSTA & ISTE. (2011). Operational definition of computational thinking. Retrieved from <https://www.iste.org/explore/Solutions/Computational-thinking-for-all>.
- Cutumisu, M., & Guo, Q. (2019). Using Topic Modeling to Extract Pre-Service Teachers' Understandings of CT From Their Coding Reflections. 62(4), *IEEE Transactions on Education*, 62(4), 325-332. <https://doi.org/10.1109/TE.2019.2925253>
- Czerkawski, B. C., & Lyman, E. W. (2015). Exploring Issues About CT in Higher Education. *TechTrends*, 59, 57–65. <https://doi.org/10.1007/s11528-015-0840-3>
- Da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, Vol. 18, pp. 17–39.
<https://doi.org/10.15388/infedu.2019.02>
- De Souza, A. A., Barcelos, T. S., Munoz, R., Villarroel, R., & Silva, L. A. (2019). Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access*, Vol. 7, pp. 82848–82866.
<https://doi.org/10.1109/ACCESS.2019.2924343>

- Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, 58(1), 240–249. <https://doi.org/10.1016/j.compedu.2011.08.006>
- Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33–39. <https://doi.org/10.1145/2998438>
- Dolgopolas, V., & Jevsikova, T. (2015). On Evaluation of CT of software engineering novice students. *Proceedings of The IFIP TC3 Working Conference “A New Culture of Learning: Computing and next Generations” At: Vilnius*, 4(2), 105–112. <https://doi.org/10.13140/RG.2.1.2855.9206>
- Dolgopolas, V., Dagienė, V., Jasutė, E., & Jevsikova, T. (2019). Design science research for computational thinking in constructionist education: A pragmatist perspective . *Problemos*, Vol. 95, pp. 144–159. <https://doi.org/10.15388/Problemos.95.12>
- Durak, H. Y., & Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers and Education*, 116, 191–202. <https://doi.org/10.1016/j.compedu.2017.09.004>
- Durak, H. Y., Yilmaz, F. G. K., & Bartın, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology*, 10(2), 173–197. <https://doi.org/10.30935/cet.554493>
- Fang, A.-D., Chen, G.-L., Cai, Z.-R., Cui, L., & Harn, L. (2017). Research on blending learning flipped class model in colleges and universities based on CT - “Database principles” for example. *Eurasia Journal of Mathematics, Science and Technology Education*, 13, 5747–5755. <https://doi.org/10.12973/eurasia.2017.01024a>

- Fernández, J. M., Zúñiga, M. E., Rosas, M. V., & Guerrero, R. A. (2018). Experiences in Learning Problem-Solving through CT. *Journal of Computer Science and Technology*, 18(02). <https://doi.org/10.24215/16666038.18.e15>
- Fletcher, G. H. L., & Lu, J. J. (2009). Education: Human computing skills: Rethinking the K-12 experience. *Communications of the ACM*, 52(2), 23–25. <https://doi.org/10.1145/1461928.1461938>
- Fraenkel, J. R., Wallen, N. E., & Hyun, H. H. (2012). *How to design and evaluate research in education* (8th ed.). Mc Graw Hill.
- Fronza, I., El Ioini, N., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education*, 17. <https://doi.org/10.1145/3055258>
- Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cárdenas, L., Pantano, P., & Bilotta, E. (2019). Lesson planning by CT skills in Italian pre-service teachers. *Informatics in Education*, 18, 69–104. <https://doi.org/10.15388/infedu.2019.04>
- García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, 80, 407–411. <https://doi.org/10.1016/j.chb.2017.12.005>
- Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: exploring computational thinking through code analysis. *Interactive Learning Environments*, Vol. 26, pp. 386–401. <https://doi.org/10.1080/10494820.2017.1337036>
- Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, 9(4), 248–260. <https://doi.org/10.1007/s00766-004-0204-6>

- Giannakos, M. N., Doukakis, S., Pappas, I. O., Adamopoulos, N., & Giannopoulou, P. (2015). Investigating teachers' confidence on technological pedagogical and content knowledge: an initial validation of TPACK scales in K-12 computing education context. *Journal of Computers in Education*, 2(1), 43–59. <https://doi.org/10.1007/s40692-014-0024-8>
- Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38–43. <https://doi.org/10.3102/0013189X12463051>
- Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, Vol. 42, pp. 38–43. <https://doi.org/10.3102/0013189X12463051>
- Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Transactions on Computing Education*, Vol. 17. <https://doi.org/10.1145/3105910>
- Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199–237. <https://doi.org/10.1080/08993408.2015.1033142>
- Günbatar, M. S. (2019). Computational thinking within the context of professional life: Change in CT skill from the viewpoint of teachers. *Education and Information Technologies*, Vol. 24, pp. 2629–2652. <https://doi.org/10.1007/s10639-019-09919-x>
- Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards CT for science majors. *SIGCSE Bulletin Inroads*, 41, 183–187. <https://doi.org/10.1145/1539024.1508931>

- Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, Vol. 27, pp. 628–644. <https://doi.org/10.1080/10494820.2019.1610451>
- Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, Vol. 17, pp. 229–244. <https://doi.org/10.15388/infedu.2018.12>
- Hooshyar, D., Malva, L., Yang, Y., Pedaste, M., Wang, M., & Lim, H. (2021). An adaptive educational computer game: Effects on students' knowledge and learning attitude in computational thinking. *Computers in Human Behavior*, 114. <https://doi.org/10.1016/j.chb.2020.106575>
- Hou, H.-Y., Agrawal, S., & Lee, C.-F. (2020). Computational thinking training with technology for non-information undergraduates. *Thinking Skills and Creativity*, 38. <https://doi.org/10.1016/j.tsc.2020.100720>
- Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, 15(9), 1277–1288. <https://doi.org/10.1177/1049732305276687>
- Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, Vol. 126, pp. 296–310. <https://doi.org/10.1016/j.compedu.2018.07.004>
- Hsu, Y.-C., Irie, N. R., & Ching, Y.-H. (2019). Computational Thinking Educational Policy Initiatives (CTEPI) Across the Globe. *TechTrends*. <https://doi.org/10.1007/s11528-019-00384-4>
- Huang, X.-P., & Leng, J. (2019). Design of database teaching model based on CT training. *International Journal of Emerging Technologies in Learning*, 14, 52–69. <https://doi.org/10.3991/ijet.v14i08.10495>

- Ioannidou, A., Bennett, V., Repenning, A., Koh, H., & Basawapatna, A. (2011). Computational Thinking Patterns Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom. Annual Meeting of the American Educational Research Association (AERA), 2. Retrieved from <http://www.agentsheets.com>
- Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, Vol. 82, pp. 263–279. <https://doi.org/10.1016/j.compedu.2014.11.022>
- Israel-Fishelson, R., & HersHKovitz, A. (2019). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research*. <https://doi.org/10.1177/0735633119887187>
- Israel-Fishelson, R., & HersHKovitz, A. (2020). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. *Journal of Educational Computing Research*, 58(5), 891–918. <https://doi.org/10.1177/0735633119887187>
- Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and CT. *Journal of Science Education and Technology*, 26, 175–192. <https://doi.org/10.1007/s10956-016-9663-z>
- Jenson, J., & Droumeva, M. (2016). Exploring media literacy and CT: A game maker curriculum study. *Electronic Journal of E-Learning*, 14, 111–121. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84968835202&partnerID=40&md5=b2d18e2de52058bda3b76bca4b55a25e>

- Jeon, Y., & Kim, T. (2017). The effects of the CT-based programming class on the computer learning attitude of non-major students in the teacher training college. *Journal of Theoretical and Applied Information Technology*, 95(17), 4330–4339.
- Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, Vol. 59, pp. 26–27.
<https://doi.org/10.1145/2955114>
- Kale, U., Akcaoglu, M., Cullen, T., & Goh, D. (2018). Contextual Factors Influencing Access to Teaching Computational Thinking. *Computers in the Schools*, Vol. 35, pp. 69–87.
<https://doi.org/10.1080/07380569.2018.1462630>
- Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal Of Modern Computing*, 4(3), 583–596.
- Kang, Y., & Lee, K. (2020). Designing technology entrepreneurship education using computational thinking. *Education and Information Technologies*, 25(6), 5357–5377.
<https://doi.org/10.1007/s10639-020-10231-2>
- Karakasis, C. and Xinogalos, S. (2020) ‘BlocklyScript: Design and Pilot Evaluation of an RPG Platform Game for Cultivating Computational Thinking Skills to Young Students’, *Informatics in Education*, 19(4), pp. 641–668. doi: 10.15388/INFEDU.2020.28.
- Katai, Z. (2020). Promoting computational thinking of both sciences- and humanities-oriented students: an instructional and motivational design perspective. *ETR&D- EDUCATIONAL TECHNOLOGY RESEARCH AND DEVELOPMENT*, 68(5), 2239–2261. <https://doi.org/10.1007/s11423-020-09766-5>
- Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing CT and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, 47, 1991–1999. <https://doi.org/10.1016/j.sbspro.2012.06.938>

- Kılıç, S., Gökoğlu, S., & Öztürk, M. (2020). A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking. *Journal of Educational Computing Research*, 073563312096440. <https://doi.org/10.1177/0735633120964402>
- Kim, Y.-M., & Kim, J.-H. (2016). Application of a software education program developed to improve computational thinking in elementary school girls. *Indian Journal of Science and Technology*, Vol. 9. <https://doi.org/10.17485/ijst/2016/v9i44/105102>
- Koehler, M. J., & Mishra, P. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge PUNYA MISHRA. *Teachers College Record*, 108(6), 1017–1054. Retrieved from http://onezoneheights.pbworks.com/f/MISHRA_PUNYA.pdf
- Kong, S.-C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, 3(4), 377–394. <https://doi.org/10.1007/s40692-016-0076-z>
- Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. In *Computers and Education* (Vol. 127, pp. 178–189). <https://doi.org/10.1016/j.compedu.2018.08.026>
- Korkmaz, Ö., & Bai, X. (2019). Adapting computational thinking scale (CTS) for chinese high school students and their thinking scale skills level. *Participatory Educational Research*, 6(1), 10–26. <https://doi.org/10.17275/per.19.2.6.1>
- Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, 72, 558–569. <https://doi.org/10.1016/j.chb.2017.01.005>

- Kukul, V., & Karataş, S. (2019). Computational thinking self-efficacy scale: Development, validity and reliability. *Informatics in Education*, Vol. 18, pp. 151–164. <https://doi.org/10.15388/infedu.2019.07>
- Kwon, J., & Kim, J. (2018). A study on the design and effect of CT and software education. *KSII Transactions on Internet and Information Systems*, 12, 4057–4071. <https://doi.org/10.3837/tiis.2018.08.028>
- Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education*, Vol. 17, pp. 321–340. <https://doi.org/10.15388/infedu.2018.17>
- Lee, Y., & Cho, J. (2020). Knowledge representation for computational thinking using knowledge discovery computing. *Information Technology and Management*, 21(1), 15–28. <https://doi.org/10.1007/s10799-019-00299-9>
- Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing Teachers to Engage Rural Students in Computational Thinking Through Robotics, Game Design, and Culturally Responsive Teaching. *Journal of Teacher Education*, Vol. 69, pp. 386–407. <https://doi.org/10.1177/0022487117732317>
- Li, M., & Hou, D. (2014). Network autonomous learning based on CT. *World Transactions on Engineering and Technology Education*, 12, 576–580. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84916237711&partnerID=40&md5=3d31af2cb32278ebb421ef9de6f7271f>
- Lin, P. H., & Chen, S. Y. (2020). Design and Evaluation of a Deep Learning Recommendation Based Augmented Reality System for Teaching Programming and CT. *IEEE Access*, 8, 45689–45699. <https://doi.org/10.1109/ACCESS.2020.2977679>

- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Lyon, J. A., & Magana, J. A. (2020). Computational thinking in higher education: A review of the literature. *COMPUTER APPLICATIONS IN ENGINEERING EDUCATION*, 28(5), 1174–1189. <https://doi.org/10.1002/cae.22295>
- Ma, J. Bin, Teng, G. F., Zhou, G. H., & Sun, C. X. (2017). Practical teaching reform on CT training for undergraduates of computer major. *Eurasia Journal of Mathematics, Science and Technology Education*, 13(10), 7121–7130. <https://doi.org/10.12973/ejmste/78738>
- Magana, A. J., & Silva Coutinho, G. (2017). Modeling and simulation practices for a CT-enabled engineering workforce. *Computer Applications in Engineering Education*, 25(1), 62–78. <https://doi.org/10.1002/cae.21779>
- Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, Vol. 80, pp. 470–477. <https://doi.org/10.1016/j.chb.2017.09.025>
- Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *BMJ (Online)*, 339(7716), 332–336. <https://doi.org/10.1136/bmj.b2535>
- Moreno León, J., Robles, G., & Román González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia*, (46), 6.
- Mousiou, M. (2021). Developing a Computational Thinking Environment through Learning Programming [Master's thesis, Hellenic Open University]. Hellenic Open University Research Repository. <https://apothesis.eap.gr/handle/repo/54054>

- Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, Vol. 33, pp. 61–76. <https://doi.org/10.14742/ajet.3521>
- Mylopoulos, J. (1992). Conceptual modelling and Telos, in: P. Loucopoulos, R. Zicari. *Conceptual Modeling, Databases, and Case An Integrated View of Information Systems Development*. Wiley New York, 1992, pp. 49–68
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*. <https://doi.org/10.1080/20004508.2019.1627844>
- Ozturk, Z., Dooley, C. M. M., & Welch, M. (2018). Finding the Hook: Computer Science Education in Elementary Contexts. *Journal of Research on Technology in Education*, 50(2), 149–163. <https://doi.org/10.1080/15391523.2018.1431573>
- Page, R., & Gamboa, R. (2013). How Computers Work: CT for Everyone. *Electronic Proceedings in Theoretical Computer Science*, 106, 1–19. <https://doi.org/10.4204/eptcs.106.1>
- Pala, F. K., & Mihçı Türker, P. (2019). The effects of different programming trainings on the CT skills. *Interactive Learning Environments*. <https://doi.org/10.1080/10494820.2019.1635495>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Park, S. Y. (2009) ‘An Analysis of the Technology Acceptance Model in Understanding University Students ’ Behavioral Intention to Use e-Learning’, *Journal of Educational Technology & Society*, 12(3), pp. 150–161.

- Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, 22(2), 421–443. <https://doi.org/10.1007/s10639-016-9475-z>
- Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic mapping studies in software engineering. 12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, 1–10.
- Piaget, J. (1970). *Genetic Epistemology*. New York: Columbia University Press.
- Pinkard, N., Martin, C. K., & Erete, S. (2019). Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments*, 0(0), 1–15. <https://doi.org/10.1080/10494820.2019.1636070>
- Qin, H. (2009). Teaching CT through bioinformatics to biology students. *SIGCSE Bulletin Inroads*, 41, 188–191. <https://doi.org/10.1145/1539024.1508932>
- Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Emerging Research, Practice, and Policy on Computational Thinking. *Emerging Research, Practice, and Policy on Computational Thinking*, 291–305. <https://doi.org/10.1007/978-3-319-52691-1>
- Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Horses, I. H. M., Basawapatna, A., & Gluck, F. (2015). Scalable Game Design. *ACM Transactions on Computing Education*, 15(2), 1–31. <https://doi.org/10.1145/2700517>
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. <https://doi.org/10.1145/1592761.1592779>
- Rodríguez-García, J. D., Moreno-León, J., Román-González, M., & Robles, G. (2020). LearningML: A tool to foster computational thinking skills through practical artificial

intelligence projects. *Revista de Educacion a Distancia*, 20(63).

<https://doi.org/10.6018/RED.410121>

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018).

Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior*, 80, 441–459.

<https://doi.org/10.1016/j.chb.2017.09.030>

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which

cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, Vol. 72, pp. 678–691.

<https://doi.org/10.1016/j.chb.2016.08.047>

Romero, M., Lepage, A., & Lille, B. (2017). CT development through creative programming

in higher education. *International Journal of Educational Technology in Higher*

Education, 14(42). <https://doi.org/10.1186/s41239-017-0080-z>

Rubinstein, A., & Chor, B. (2014). CT in Life Science Education. *PLoS Computational*

Biology, 10. <https://doi.org/10.1371/journal.pcbi.1003897>

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming

languages integrated across the curriculum in elementary school: A two year case study using “scratch” in five schools. *Computers and Education*, 97, 129–141.

<https://doi.org/10.1016/j.compedu.2016.03.003>

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

2013, 5–8.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating

computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, Vol. 18, pp. 351–380.

<https://doi.org/10.1007/s10639-012-9240-x>

- Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2), 469–495. <https://doi.org/10.1007/s10639-016-9482-0>
- Shih, H., Jackson, J. M., Hawkins-Wilson, C. L., & Yuan, P.-C. (2015). Promoting CT skills in an emergency management class with MIT app inventor. *Computers in Education Journal*, 6, 82–91. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85052799029&partnerID=40&md5=df2b3b5a13cf93ab789a443cd5d7dd>
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, Vol. 22, pp. 142–158. <https://doi.org/10.1016/j.edurev.2017.09.003>
- Siau, K., & Tan, X. (2005). Improving the quality of conceptual modeling using cognitive mapping techniques. *Data and Knowledge Engineering*, 55(3), 343–365. <https://doi.org/10.1016/j.datak.2004.12.006>
- Snow, E., Rutstein, D., Basu, S., Bienkowski, M., & Everson, H. T. (2019). Leveraging Evidence-Centered Design to Develop Assessments of Computational Thinking Practices. *International Journal of Testing*, Vol. 19, pp. 103–127. <https://doi.org/10.1080/15305058.2018.1543311>
- Sondakh, D. E., Osman, K., & Zainudin, S. (2020). A proposal for holistic assessment of computational thinking for undergraduate: Content validity. *European Journal of Educational Research*, 9(1), 33–50. <https://doi.org/10.12973/eu-jer.9.1.33>
- Souza, M. R. d. A., Veado, L., Moreira, R. T., Figueiredo, E., & Costa, H. (2018). A systematic mapping study on game-related methods for software engineering education. *Information and Software Technology*, 95, 201–218. <https://doi.org/10.1016/j.infsof.2017.09.014>

- Sun, L., Hu, L., & Zhou, D. (2022). Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. *Computers and Education*, 181(27), 104457. <https://doi.org/10.1016/j.compedu.2022.104457>
- Taylor, N. G., Moore, J., Visser, M., & Drouillard, C. (2018). Incorporating CT into library graduate course goals and objectives. *School Library Research*, 21. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85053260681&partnerID=40&md5=15815233f96a5912185346fe719f6496>
- Tikva, C. & Tambouris, E. (2021a) 'A systematic mapping study on teaching and learning Computational Thinking through programming in higher education', *Thinking Skills and Creativity*, 41(December 2020), p. 100849. doi: 10.1016/j.tsc.2021.100849.
- Tikva, C., & Tambouris, E. (2021b). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, 162, 104083. <https://doi.org/10.1016/j.compedu.2020.104083>
- von Wangenheim, C. G., Hauck, J. C. R., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F. (2018). CodeMaster - Automatic assessment and grading of app inventor and snap! Programs. *Informatics in Education*, 17(1), 117–150. <https://doi.org/10.15388/infedu.2018.08>
- Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.
- Wand, Y., & Weber, R. (2002). Research commentary: Information systems and conceptual modeling - A research agenda. *Information Systems Research*. <https://doi.org/10.1287/isre.13.4.363.69>
- Webster, J., & Watson, R. (2002). Analyzing the past to prepare for the future: writing a literature review. *MIS Quarterly*, 26(2), 13–23.

- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, Vol. 25, pp. 127–147.
<https://doi.org/10.1007/s10956-015-9581-5>
- Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning*, Vol. 6, pp. 1–17. <https://doi.org/10.4018/IJGBL.2016010101>
- Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. SIGCSE'12 - proceedings of the 43rd ACM technical symposium on computer science education. <https://doi.org/10.1145/2157136.2157200>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725. <https://doi.org/10.1098/rsta.2008.0118>
- Witherspoon, E. B., & Schunn, C. D. (2019). Teachers' goals predict computational thinking gains in robotics. *Information and Learning Science*, 120(5–6), 308–326. <https://doi.org/10.1108/ILS-05-2018-0035>
- Wu, B., Hu, Y., Ruis, A. R., & Wang, M. (2019). Analysing CT in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning*, 35, 421–434. <https://doi.org/10.1111/jcal.12348>
- Xing, W. (2019). Large-scale path modeling of remixing to computational thinking. *Interactive Learning Environments*. <https://doi.org/10.1080/10494820.2019.1573199>

- Yadav, A., Mayfield, C., Zhou, N., Hambruch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, Vol. 14. <https://doi.org/10.1145/2576872>
- Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, Vol. 60, pp. 55–62. <https://doi.org/10.1145/2994591>
- Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, 24(1), 929–951. <https://doi.org/10.1007/s10639-018-9801-8>
- Yuen, T. T., & Robbins, K. A. (2014). A qualitative study of students' CT skills in a data-driven computing class. *ACM Transactions on Computing Education*, 14. <https://doi.org/10.1145/2676660>
- Zha, S., Jin, Y., Moore, P., & Gaston, J. (2020a). A cross-institutional investigation of a flipped module on preservice teachers' interest in teaching computational thinking. *Journal of Digital Learning in Teacher Education*, 36(1), 32–45. <https://doi.org/10.1080/21532974.2019.1693941>
- Zha, S., Jin, Y., Moore, P., & Gaston, J. (2020b). Hopscotch into Coding: Introducing Pre-Service Teachers Computational Thinking. *TechTrends*, 64(1), 17–28. <https://doi.org/10.1007/s11528-019-00423-0>
- Zhang, L. C. & Nouri, J. (2019) 'A systematic review of learning computational thinking through Scratch in K-9', *Computers and Education*, 141(June), p. 103607. doi: 10.1016/j.compedu.2019.103607.
- Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers and Education*, 141(July), 103633. <https://doi.org/10.1016/j.compedu.2019.103633>

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