9

LEARNING COMPUTATIONAL THINKING IN CO-CREATION PROJECTS WITH MODERN EDUCATIONAL TECHNOLOGY: EXPERIENCES FROM FINLAND

Kati Mäkitalo, Matti Tedre, Jari Laru, Teemu Valtonen, Megumi Iwata, and Jussi Koivisto

INTRODUCTION

Living and working in a digitalized society requires skills and competences that are often referred to as twenty-first-century skills. One of those new skills, computational thinking (CT), has recently become a catchphrase for a broad variety of efforts to bring computing skills and knowledge into the school curriculum. Computational thinking is seen not only as a part of computer science courses but as a part of all school subjects (Guzdial 2015; Tedre and Denning 2016; Mouza et al. 2017; Lockwood and Mooney 2017; Denning and Tedre 2019). Computational thinking promises the skills and competences necessary for understanding, controlling, and automating information processes as well as for interpreting the world as information processes (Denning and Tedre 2019, 4). The introduction of CT to K–12 schools, curriculum guidelines, and frameworks have been prepared by major organizations, including the Computer Science Teachers Association (CSTA) in the US, Computing at School (CAS) in the UK, the Australian Curriculum, Assessment and Reporting Authority (ACARA), as well a large number of other national bodies around the world. The Finnish national basic education core curricula (The National Core Curriculum 2014) now include computing—although not as a separate subject to be taught but as a cross-curricular topic that should be integrated in all subjects.

After a long history in academia and the school system (Denning and Tedre 2019, 175–192), the latest wave of CT for K–12 started in 2006 when Jeannette Wing leveraged her position at the US National Science Foundation (NSF) to campaign her CT vision to the public, funding bodies, and academic audiences (Wing 2006). Wing persuaded decision-makers to intensify CT efforts in schools in many countries, starting with the US. Wing's efforts in NSF directorship led to some impressive achievements, such as a new Advanced Placement (AP) program in the US, billions of dollars invested by the Obama administration in CS for All (NSF 2016), the training of ten thousand computing teachers in the US, and a surge of popular initiatives like code.org, k12cs.org, and Google for Education. The latest literature surveys list hundreds of recent articles, textbooks, and journal special issues (Lockwood and Mooney 2017; García-Peñalvo et al. 2016). We can find several early models and tools for testing CT skills, such as the Mobile CT model (Sherman and Martin 2015), the Progression of Early CT model (Seiter and Foreman 2013), and the Real-Time Evaluation and Assessment of CT tool (Koh et al. 2014). Dagienė's CT-related "Bebras Challenge" has been utilized worldwide by more than two million learners (bebraschallenge.org). The International Computer and Information Literacy Study (ICILS) has included CT in its evaluation since 2018 (Bundsgaard et al. 2019). Despite all this, there is a consensus in the "CT for K–12" community that there are remarkable gaps in the state-of-the-art knowledge about CT in K–12 schools. The research literature has repeatedly highlighted three critical problems.

The first problem, which Denning called one of the "remaining trouble spots" with CT education, is: "How do we measure learners' computational abilities?" (Denning 2017). For example, Mark Guzdial's (2015) quintessential textbook on computing education barely touches the topic of how to evaluate CT skills. A 2017 literature review of more than two hundred studies noted that despite some models for testing CT skills, work for testing CT and the existing models are at their early stages of development (Lockwood and Mooney 2017). One of the reports identified a few pioneering articles, each of which urged researchers to turn their attention to how to measure CT skill development (Mannila et al. 2014). Those problems are due to a lack of consensus over what skills CT consists of and what skill progression in CT looks like.

The second problem is: "How do we integrate CT in K–12 subjects?" Typically, CT is taught in specialized computing courses rather than being integrated in school subjects. While there is less empirical research on whether and how the synergies from integrating CT into K–12 subjects promotes students' analytical skills or widens their understanding about CT (Guzdial 2015, 41–51), integrating CT in subjects is crucial for understanding the ways in which digitalization and computerization have changed all sciences and areas of life. In line with the educational vision of teaching disciplinary ways of thinking and practicing, the knowledge and skills of CT fit well the formal learning goals of other subjects. However, due to fewer concrete examples of how to integrate CT into K–12 subjects, there is a need for empirical evidence on how CT integration influences learners' learning outcomes. Promoting CT in K–12 is challenging due to its specialized nature and broad applicability. Few K–12 teachers have the requisite knowledge and skills of CT as a concept or in its integration in other subjects, and while computer scientists have knowledge and skills in CT in computing, they lack the knowledge and skills for CT integration in education (Mouza et al. 2017).

The third critical problem has to do with lack of understanding: "How do teachers learn to synthesize CT with existing content and pedagogical strategies?" (cf. Mouza et al. 2017). We need to understand the technological knowledge and skills needed for teachers to integrate technology and technological resources effectively in classrooms. We have examples of studies on how teachers combine the areas of technology and pedagogy with the content taught (see Koehler and Mishra 2009). Valtonen et al. (2019) show that despite intensive integration of technology, content, and pedagogy, pre-service teachers experience technological knowledge as a separate domain. Studies show that integrating CT in teacher education courses enhances pre-service teachers' knowledge and understanding of CT, but when CT is taught separately from teachers' own disciplines, that understanding remains at an abstract level and is not applied in teaching (Yadav, Stephenson, and Hong 2017). We can create a framework that offers a practical model for integrating CT within those subject matters and pedagogical approaches that teachers are expected to teach in classrooms (Yadav et al. 2017). However, CT literature has pointed out a dire need for more research on understanding

teacher learning and the best ways to support it (Guzdial 2015; Mouza et al. 2017).

This chapter presents two example cases to broaden our understanding of how learning CT ideas can be supported through technology and classroom practices in the context of STEAM topics and what leeway CT gives to STEAM classes. It also presents what one can learn from nonformal education in an after-school programming club. We investigate learners' as well teachers' CT knowledge and skills through the cases provided here.

CASE 1: PROMOTING COMPUTATIONAL THINKING WITH MINECRAFT IN A K–12 PROGRAMMING CLUB

This case study was designed to integrate informal learning activities and computational thinking concepts for learners in the context of an after-school Minecraft club. The design rationale for the Earth 2.0 Minecraft game was recurring difficulties in finding a functional and motivating way to teach computational thinking and programming in an after-school club (Koivisto, Laru, and Mäkitalo 2019).

To solve this issue, a pedagogically and theoretically grounded Minecraft game learning experience was designed and evaluated with the club participants in an authentic setting. Minecraft is a multiplayer sandbox game designed around breaking and placing blocks. Unlike many other games, when played in its traditional settings, Minecraft grants players the freedom to immerse themselves into their own narrative: to build, create, and explore. Minecraft, along with modification software ("mods"), has the tools for teaching and learning programming (Zorn et al. 2013; Risberg 2015; Nebel, Schneider, and Rey 2016; Näykki et. al 2019).

TOOLS

The tools used were the Minecraft game, Earth 2.0 map, and seven Minecraft modifications (mods). Earth 2.0 includes problem-based puzzles embedded in an engaging post-apocalyptic narrative. Modifications enable one to modify Minecraft's eighteen game rules, alter game content, redesign textures, and give players new abilities (Kuhn and Dikkers 2015). While Earth 2.0 provided a context for the game narrative and

gameplay itself, modifications worked as an engine to enable interactivity and programming during the game.

The most important modification in this context was ComputerCraftEdu. It is an extension based on Dan200's ComputerCraft but with some added features, including the programmable Beginners Turtle (see table 9.1), which learners are able to program with descriptive icons. Otherwise, learners must know the text-based LUA language, which is used in both extensions. According to Wilkinson, Williams, and Armstrong (2013), the use of ComputerCraft has a positive impact on learning and motivation in programming. All mods used in this experiment are briefly explained in table 9.1.

PARTICIPANTS

Participant groups enrolled in after-school programming clubs for three months with the first four sessions on Minecraft. The data were collected from five clubs, each consisting of eight to twenty participants between seven and twelve years of age, with altogether sixty-two participants (nine female and fifty-three male). All five groups used the prototype version of the Earth 2.0 computational thinking game[1] (Koivisto, Laru, and Mäkitalo 2019).

Table 9.1 All the mods used to expand original Minecraft gameplay

Modification	Purpose of the modification in Earth 2.0
Forge	Minecraft modding API
ComputerCraft	Virtual computers for programming scripted events
ComputerCraftEdu	Educational version of ComputerCraft that includes the Beginners Turtle.
CustomNPCs	Characters to make interaction between player and the game world possible and meaningful
Malisisdoors and Malisiscore	Code-locked and animated doors
Optifine	Minecraft's optimization to improve performance
JammerCraft	More realistic and detailed textures that match with the narrative of Earth 2.0

TASKS AND PEDAGOGICAL DESIGN

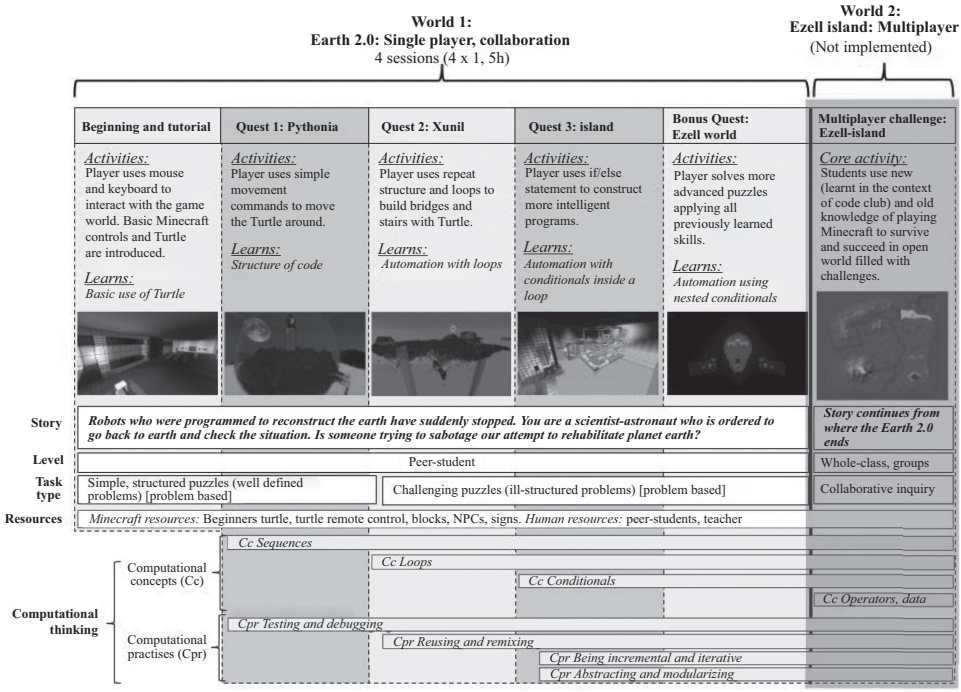
In the Earth 2.0 Minecraft world, the game narrative starts from a problem the players face. Players should program robots to reconstruct the post-apocalyptic Earth, but suddenly all activities stop. Players are assigned the role of scientist-astronaut, and they are instructed to go back to the Earth and investigate the situation. On Earth, they find information about a person who is trying to sabotage the scientist-astronauts' attempts to rehabilitate the planet Earth.

The game narrative is further divided into four main puzzles (quests) and one bonus puzzle (quest), which all are separate Minecraft worlds and one bonus world. First, players must take the tutorial to learn to use the programming tool, Beginners Turtle. Then they are introduced to basic programming concepts and practices in a specific order (see table 9.2). Concepts and practices of computational thinking are distributed to three sequential task groups: Quests 1, 2, and 3. Finally, a fourth task group, Bonus Quest, is included for more advanced players (see detailed descriptions in figure 9.1 and table 9.2). The layout of an individual puzzle is presented in figures 9.2 and 9.3.

The Earth 2.0 experiment in the context of the programming club is an example of using off-shelf sandbox games as a tool to teach CT concepts and practices for inexperienced learners in the K–12 context. According to Lye and Koh (2014), it is dubious to assume that learners could figure out computational practices and perspectives through pure self-discovery. Support from more experienced learners and teachers helps with understanding better computational practices. So, it is necessary to bring all participants on the same level of the basic ideas of programming and computational thinking by using different scaffolds that restrict original gameplay, although it violates the core principles and ideas of the constructivist gameplay (Kafai and Burke 2015; Lye and Koh 2014; Mayer 2015).

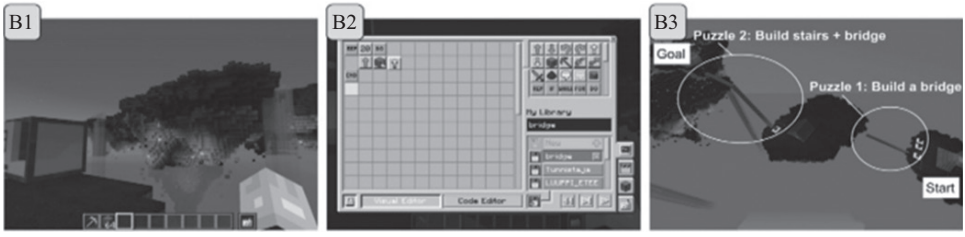
CASE 2: K–12 TEACHERS' AND FAB LAB FACILITATORS' ATTITUDES AND BELIEFS ABOUT COMPUTATIONAL THINKING IN DIGITAL FABRICATION AT FAB LAB

In this case study, the aim was to explore learners' digital fabrication activities at Fab Lab Oulu, located in the University of Oulu, Finland. The



9.1 The pedagogical design, where computational concepts and practices were integrated as a part of the Minecraft gameplay.

9.2 Quest 2, puzzle 1. A1: Player encounters a problem. A2: Player programs the Turtle to build a bridge. A3: Player executes the program, and Turtle builds a bridge for the player.



9.3 Quest 2, puzzle 2. B1: Player encounters a slightly different problem. B2: Player remixes the bridge program to build stairs. B3: Overview of puzzles 1–2 in Quest 2, both puzzles completed.

Table 9.2 Tasks to promote computational thinking included in Earth 2.0

Game phase	Task	Structures and statements used
Tutorial	Practice using the (Beginners) Turtle to open a door to the portal room (to advance to Quest 1).	Only movement command (single instruction)
Quest 1: puzzles 1–3	Use movement commands to move the Turtle and advance.	Only movement commands (sequential execution)
Quest 1: puzzle 4	Use loop to move the Turtle long distances.	While-true-do (iteration)
Quest 2, puzzle 1	Build a bridge.	"Repeat" structure (iteration)
Quest 2, puzzle 2	Remix bridge program to build stairs followed by a bridge.	"Repeat" structure (iteration)
Quest 2, puzzle 3	Use loop to move the Turtle long distances.	While-true-do (iteration)
Quest 2, puzzle 4	Remix bridge program to repair broken bridge.	"Repeat" structure (iteration)
Quest 3, puzzle 1	Practice to use conditionals and loops together to dig through a cave.	While-true-do, (iteration) If-then-else (conditional branching / selection)
Quest 3, puzzle 2	Program automated Turtle to avoid obstacles.	While-true-do, (iteration), if-then-else (selection)
Quest 3, puzzle 3	Remix previous program to build a bridge while avoiding obstacles.	While-true-do, (iteration), if-then-else (selection)
Quest 3, puzzle 4	Program automated Turtle to solve a labyrinth.	While-true-do, if-then-else if
Quest 3, puzzle 5	Reuse and debug previous program to solve bigger labyrinth.	While-true-do, if-then-else if
Quest 3, puzzle 6	Remix bridge, obstacle, and digger programs to reach the exit.	While-true-do, if-then-else if
Bonus Quest, puzzle 1	Reuse bridge program and stair program to create path to the exit.	Repeat-structure, while-true-do
Bonus Quest, puzzle 2	Create intelligent bridge-builder program to create path.	While-true-do, if-then-else if

Fab Lab offers digital fabrication facilities for K–12 schools in the Oulu area. Those activities typically combine 2D and 3D designing and manufacturing, prototyping with electronics, basic embedded systems programming, and using various manufacturing tools and machines. Digital fabrication activities combine content and skills from multiple subjects, and the activities are typically ill structured and student centered (Iwata et al. 2020; Näykki et al. 2019; Pitkänen, Iwata, and Laru 2020).

In this case study, seventh- to ninth-grade learners from three K–12 schools in Oulu participated in digital fabrication activities that spanned from three to five days at Fab Lab Oulu (see table 9.3). Learners worked on digital fabrication projects in groups of two to five. Activities were led by the Fab Lab instructors and were observed by teachers. The activities were

Table 9.3 Participants and project details of three case schools

Participants Project details	Case I: School A	Case II: School B	Case III: School C
Number of participants	12 (15–16 years), 1 teacher	20 (13–15 years), 2 teachers	9 (15–16 years), 2 teachers
Duration	5 days	3 days	5 days
Projects	Useless box, Rail for camera, Electronic controlled lock, Jukebox game, Music car	Finland's 100-year anniversary calendar, Finland 100-years history wheel, Finnish flag day clock	Two models of playhouse
Required conditions	Use Arduino Uno board and at least one actuator, fabricate mechanics using laser cutter or 3D printer, make functional artefact	Use Arduino Uno board, fabricate mechanics using laser cutter	A competition between two teams designing a playhouse for the school community
Software for designing	Inkscape, Autodesk TinkerCad	Inkscape	Inkscape, SketchUp
Machines	Laser cutter, 3D printer	Laser cutter	Laser cutter, vinyl cutter, sewing machine
Electronics	Arduino Uno board, servos, buttons, piezoelectric buzzer	Arduino Uno board, servos	–
Programming	Arduino Software (IDE)	Arduino Software (IDE)	–

loosely structured with minimal instruction, and the activity designs differed among schools. The instructors gave assistance only when learners had problems in the process. Two focus group interviews were conducted after the digital fabrication activities. Three teachers from two schools (focus group interview I) and two instructors at the Fab Lab (focus group interview II) participated in the focus group interviews.

In the focus group interview, teachers described the ways digital fabrication activities make an effective context for integrating CT with knowledge and skills from multiple subjects. One of the teachers revealed the following: "this kind of working [project-based digital fabrication activities] brings together very many school subjects. There are mathematics, physics, coding, programming." That teacher specifically linked programming with the learning goals of the national core curriculum for basic education.

Teachers explained the benefits of tangible project-based activities in relation to learning CT: Learning CT becomes more effective when learners have concrete projects where they can apply knowledge into practice. When learner groups had challenges using the microcontroller, the instructor gave short lectures on how the logical ports of the microcontroller work. One teacher described that those lectures helped learners develop an understanding of how computers work, which they can utilize in their projects on the fly. In project-based activities, learners were able to test their CT knowledge in real projects.

Teachers found that the digital fabrication activities that combine hardware and software into one project are effective for learning CT. They felt that the processes of making physical objects enhanced learners' logical thinking (their projects made use of logic gates). Although learners have a general and abstract idea of the processes of making a physical object, they often confront many practical challenges during digital fabrication activities. When dealing with physical parts, learners need to think in concrete, sequential, and interconnected steps to complete their digital fabrication projects. Teachers reflected that "when we started, there was already the idea that it's going to be a process of several steps . . . they [the learners] just didn't know what's going to happen in between those steps." This teacher stated that previously he thought combining hardware and programming is challenging, and therefore he

gave ready-made parts that learners only needed to assemble. After visiting the Fab Lab, he shifted his perspective, explaining that in the future learners have the responsibility to plan, design, and create the hardware parts by themselves. In the Fab Lab, teachers observed learners' activities and their performance, and, based on the observations, he decided to change his teaching practice.

However, the same case study also exposed challenges for learning CT in digital fabrication activities (see table 9.4). Firstly, the teachers' discussion in the focus group regarding CT was still at the surface level. During the focus group interview, teachers mentioned several aspects of

Table 9.4 CT practices identified in focus group interviews

	Focus group interview I (*N* = 8,387)		**Focus group interview II (*N* = 6,328)**	
CT practices(Barr, Harrison, and Conery 2011)	%	N	%	n
1) Formulating problems in a way that computers and other tools can help solve them	45.8%	432	17.8%	147
2) Logically organizing and analyzing data	18.1%	171	28.9%	239
3) Representing data through abstractions	0.0%	–	0.0%	–
4) Automating solutions through algorithmic thinking	7.1%	67	17.8%	147
5) Identifying, analyzing, and implementing possible solutions with the most efficient and effective combination	29.0%	274	35.5%	293
6) Generalizing and transferring this problem-solving process	0.0%	–	0.0%	–
Total	100%	944	100%	826

Note: N shows the total number of words in the focus group interview; n shows the number of words at the node.

CT. These aspects can be categorized based on Barr et al.'s work (2011, 21), which are (1) "formulating problems in a way that computer and other tools can help solve them," (2) "identifying, analyzing, and implementing possible solutions with the most efficient and effective combination," and (3) "logically organizing and analyzing data." Nevertheless, none of those CT aspects were discussed more deeply in the focus group interviews. The instructors who facilitated the digital fabrication activities mentioned that K–12 teachers are unfamiliar with computational concepts and practices as they are not embedded in the curriculum and school culture (Iwata et al. 2020).

Secondly, teachers indicate needs for pedagogy in ill-structured digital fabrication activities to enhance learning (Pitkänen, Iwata, and Laru 2020). Currently, K–12 teachers are not actively involved in the preparation and implementation processes of activities at the Fab Lab, which require technical knowledge and skills beyond teachers' competences. Low involvement might be due to Fab Lab activities, which are not embedded in the curriculum. This case study illustrates that there is a need to involve teachers more in these kinds of activities to enable them to integrate CT with pedagogical approaches as well as assessing CT based on the goals of the curriculum.

DISCUSSION

Our case studies are a part of a larger attempt at integrating CT into Finnish education. They are also aimed at broadening the current understanding of twenty-first-century teacher education by producing empirical results on the effect of Fab Lab interventions on teachers' CT knowledge and skills and using games as a learning environment for CT skills.

The Minecraft example (first case study) demonstrates how CT ideas can be embedded in a constructionist computer game. However, if concepts and practices of CT are not explicit, visible, and measurable, learning CT remains at the surface level for both learners and teachers. Minecraft is an immersive 3D game, which helps teachers and learners to design experiences in which concepts of CT can be very visible and explicit. In that first case study, to complete each quest (challenges), the learner's task was to program Beginner's Turtle to, for example, build a bridge.

During this activity, learners learn both the concepts (e.g., repeat structures) and the practices (e.g., remixing and debugging). Microsoft has adopted a similar approach in its Minecraft Education Edition "Hour-of-Code" tutorial,[2] which is oriented to beginners like the world in the first case study was. However, Minecraft would also be a good platform to teach a more holistic approach to CT than these very structured and scaffolded tutorials oriented to teaching coding skills. Next, we should consider what kind of support mechanisms for helping teachers and students could be designed for adopting a holistic approach to CT concepts, skills, and practices in Minecraft.

The second case study explored K–12 teachers' and Fab Lab facilitators' attitudes and beliefs about CT in the context of digital fabrication. Ill-structured digital fabrication activity, which involves multiple subject matters, is complex context, but it provides a good opportunity for the integration of CT in the school curriculum. Project-based activities where learners deal with hardware parts, components, and software, require, for instance, decomposing complex problems into subproblems and logically organizing solutions into steps to achieve the goals.

However, this second case study indicates the need for further developing teachers' CT understanding, especially in terms of integrating CT in the curriculum with appropriate pedagogical approaches. It was found that teachers and facilitators are not fully aware of the concepts of CT. Teachers' understanding covered only the surface of CT, such as using computers to solve problems, while CT practices, which involve the fundamental concepts of CT, such as abstractions and automation, were not intensively discussed (Iwata et al. 2020).

On the other hand, facilitators are already adept at CT skills, and they may not have the awareness of defining CT to be a competence that learners need to develop. Teachers' and facilitators' awareness of the concepts of CT may be essential to providing opportunities for learners to understand and apply CT practices in digital fabrication activities. Insufficient understanding of CT concepts and practices might diminish the potential to develop CT in digital fabrication activities (Iwata et al. 2020).

The findings are in line with Mannila et al. (2014, 14), illustrating that very few teachers implement the concepts of abstraction and automation in classroom practice. Furthermore, other studies have shown that K–12

teachers' understanding about CT competence and skills often remains at a superficial level, typically at the level of basic programming skills (Mouza et al. 2017). Yet, that result is unsurprising in the absence of consensus over even the most fundamental principles of CT in curricular integration (Denning and Tedre 2019). Increased scaffolding efforts turned out to benefit learning, especially before engaging in complex ill-structured problem-solving activities in the context of STEAM projects or in digital fabrication and making (Fab Lab). The question is: Do we need better pedagogical design and increased scaffolding when we try to introduce CT concepts, practices, and perspectives to K–12 education? Do we have adequate pre- and in-service teacher education that would cover these emerging topics?

From the Finnish perspective, the ideas of CT are embedded in our national core curriculum for basic education without naming CT. For instance, key content areas related to the objectives of mathematics in grades 7–9 include: (1) thinking skills and methods, such as logical thinking and discovering rules and dependencies, (2) examining and applying functions, and (3) data processing, such as collecting, structuring, and analyzing data (Finnish National Board of Education 2016, 402–405). Connecting those mentioned skills with problem-solving, which applies how computers work, enhance teachers' understanding of holistic views of CT (Denning and Tedre 2019, xi).

However, a few elements need to come together for better curricular integration of CT in Finland and other countries. First, we need to build a common understanding about the fundamental computational principles necessary for education. Second, we need working models, templates, and exemplars for integrating CT in curricula. Third, we need pedagogical models that make CT competences and skills more visible. Fourth, we need principles for assessing CT within subjects as well as based on the goals of the basic education curriculum. Instead of a technology-driven or programming-driven approach for integrating CT in education, we need a human- and design-driven as well discipline-driven approach to understanding CT in a broader way.

ACKNOWLEDGMENTS

Thank you to the Fab Lab facilitators, participating teachers and learners, and participants and facilitators of the after-school programming clubs.

NOTES

1. Description of Earth 2.0 Minecraft game: https://sites.google.com/oulu.fi/earth20/english.
2. https://education.minecraft.net/hour-of-code.

REFERENCES

Barr, David, John Harrison, and Leslie Conery. 2011. "Computational Thinking: A Digital Age Skill for Everyone." *Learning & Leading with Technology* 38, no. 6: 20–23.

Bundsgaard, Jeppe, Sofie Gry Bindslev, Elisa Nadire Caeli, Morten Pettersson, and Anna Rusmann. 2019. *Danske elevers teknologiforståelse: Resultater fra ICILS-undersøgelsen 2018*. Aarhus, Denmark: Aarhus Universitetsforlag.

Denning, Peter J. 2017. "Remaining Trouble Spots with Computational Thinking." *Communications of the ACM* 60, no. 6: 33–39.

Denning, Peter J., and Matti Tedre. 2019. *Computational Thinking*. Cambridge, MA: MIT Press.

Finnish National Board of Education. 2016. *National Core Curriculum for Basic Education 2014*. Helsinki: Finnish National Board of Education.

García Peñalvo, Francisco José, Daniela Reimann, Maire Tuul, Alyson Rees, and Ilkka Jormanainen. 2016. "An Overview of the Most Relevant Literature on Coding and Computational Thinking with Emphasis on the Relevant Issues for Teachers." Technical Report, TACCLE3 Consortium, Belgium.

Guzdial, Mark. 2015. "Learner-Centered Design of Computing Education: Research on Computing for Everyone." *Synthesis Lectures on Human-Centered Informatics* 8, no. 6: 1–165. San Rafael, CA: Morgan & Claypool.

Iwata, Megumi, Kati Pitkänen, Jari Laru, and Kati Mäkitalo. 2020. "Exploring Potentials and Challenges to Develop Twenty-First Century Skills and Computational Thinking in K-12 Maker Education." *Frontiers in Education* 5: 87. https://doi.org/10.3389/feduc.2020.00087.

Kafai, Yasmin B., and Quinn Burke. 2015. "Constructionist Gaming: Understanding the Benefits of Making Games for Learning." *Educational Psychologist* 50, no. 4: 313–334.

Koehler, Matthew, and Punya Mishra. 2009. "What Is Technological Pedagogical Content Knowledge (TPACK)?" *Contemporary Issues in Technology and Teacher Education* 9, no. 1: 60–70.

Koh, Kyu Han, Ashok Basawapatna, Hilarie Nickerson, and Alexander Repenning. 2014. "Real Time Assessment of Computational Thinking." In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*: 49–52. IEEE.

Koivisto, Jussi, Jari Laru, and Kati Mäkitalo. 2019. "Promoting Computational Thinking Skills in the Context of Programming Club for K-12 Pupils with the Engaging Game Adventure in Minecraft." In *Proceedings of International Conference on Computational Thinking Education 2019*, 48. Hong Kong: Education University of Hong Kong.

Kuhn, Jeff, and Seann Dikkers. 2015. "How Can Third Party Tools Be Used?" In *Teachercraft: How Teachers Learn to Use Minecraft in Their Classrooms*, 123–138. Carnegie Mellon University Pittsburgh: ETC Press.

Lockwood, James, and Aidan Mooney. 2017. "Computational Thinking in Education: Where Does It Fit? A Systematic Literary Review." *arXiv preprint arXiv:1703.07659*. Technical report, National University of Ireland Maynooth.

Lye, Sze Yee, and Joyce Hwee Ling Koh. 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.

Mannila, Linda, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, and Amber Settle. 2014. "Computational Thinking in K-9 Education." In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*: 1–29.

Mayer, Richard E. 2015. "On the Need for Research Evidence to Guide the Design of Computer Games for Learning." *Educational Psychologist* 50, no. 4: 349–353. https://doi.org/10.1080/00461520.2015.1133307.

Mouza, Chrystalla, Hui Yang, Yi-Cheng Pan, Sule Yilmaz Ozden, and Lori Pollock. 2017. "Resetting Educational Technology Coursework for Pre-Service Teachers: A Computational Thinking Approach to the Development of Technological Pedagogical Content Knowledge (TPACK)." *Australasian Journal of Educational Technology* 33, no. 3: 61–76.

Näykki, Piia, Jari Laru, Essi Vuopala, Pirkko Siklander, and Sanna Järvelä. 2019. "Affective Learning in Digital Education—Case Studies of Social Networking Systems, Games for Learning and Digital Fabrication." *Frontiers in Education* 4 (November): 128. https://doi.org/10.3389/feduc.2019.00128.

Nebel, Steve, Sascha Schneider, and Günter Daniel Rey. 2016. "Mining Learning and Crafting Scientific Experiments: A Literature Review on the Use of Minecraft in Education and Research." *Journal of Educational Technology & Society* 19, no. 2: 355–366.

NSF [National Science Foundation]. 2016. "CS for All." Accessed November 23, 2020. https://www.nsf.gov/news/special_reports/csed/csforall.jsp.

Pitkänen, Kati, Megumi Iwata, and Jari Laru. 2020. "Exploring Technology-Oriented Fab Lab Facilitators' Role as Educators in K-12 Education: Focus on Scaffolding

Novice Students' Learning in Digital Fabrication Activities." *International Journal of Child-Computer Interaction* 26. https://doi.org/10.1016/j.ijcci.2020.100207.

Risberg, Cathy. 2015. "More than Just a Video Game: Tips for Using Minecraft to Personalize the Curriculum and Promote Creativity, Collaboration, and Problem Solving." *Illinois Association for Gifted Children Journal*, 44–48. Accessed November 26, 2020. https://studylib.net/doc/14185247/iagc-journal-focus--creativity--critical-thinking-.

Seiter, Linda, and Brendan Foreman. 2013. "Modeling the Learning Progressions of Computational Thinking of Primary Grade Students." In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, 59–66. ACM.

Sherman, Mark, and Fred Martin. 2015. "The Assessment of Mobile Computational Thinking." *Journal of Computing Sciences in Colleges* 30, no. 6: 53–59.

Tedre, Matti, and Peter J. Denning. 2016. "The Long Quest for Computational Thinking." In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, 120–129. ACM.

The National Core Curriculum. 2014. "The Finnish National Board of Education." Accessed September 9, 2018. https://www.oph.fi/fi/koulutus-ja-tutkinnot/perusopetuksen-opetussuunnitelman-perusteet.

Valtonen, Teemu, Erkko Sointu, Jari Kukkonen, Kati Mäkitalo, Nhi Hoang, Päivi Häkkinen, Sanna Järvelä et al. 2019. "Examining Pre-Service Teachers' Technological Pedagogical Content Knowledge as Evolving Knowledge Domains: A Longitudinal Approach." *Journal of Computer Assisted Learning* 35, no. 4: 491–502.

Wilkinson, Brett, Neville Williams, and Patrick Armstrong. 2013. "Improving Student Understanding, Application and Synthesis of Computer Programming Concepts with Minecraft." In *The European Conference on Technology in the Classroom*.

Wing, Jeannette M. 2006. "Computational Thinking." *Communications of the ACM* 49, no. 3: 33–35.

Yadav, Aman, Chris Stephenson, and Hai Hong. 2017. "Computational Thinking for Teacher Education" *Communications of the ACM* 60, no. 4: 55–62.

Zorn, Christopher, Chadwick A. Wingrave, Emiko Charbonneau, and Joseph J. LaViola, Jr. 2013. "Exploring Minecraft as a Conduit for Increasing Interest in Programming." In *Proceedings of the 8th International Conference on the Foundations of Digital Games (FDG 2013)*, 352–359. Chania, Crete, Greece. http://www.fdg2013.org/program/papers/paper46_zorn_etal.pdf.

This is a section of doi:10.7551/mitpress/14041.001.0001

Computational Thinking Curricula in K–12

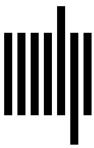
International Implementations

Edited by: Harold Abelson, Siu-Cheung Kong

Citation:

Computational Thinking Curricula in K–12: International Implementations
Edited by: Harold Abelson, Siu-Cheung Kong
DOI: 10.7551/mitpress/14041.001.0001
ISBN (electronic): 9780262378642
Publisher: The MIT Press
Published: 2024

The open access edition of this book was made possible by generous funding and support from MIT Press Direct to Open



The MIT Press

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Stone Serif by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Abelson, Harold, editor. | Kong, Siu Cheung, editor.
Title: Computational thinking curricula in K-12 : international implementations / edited by Harold Abelson and Siu-Cheung Kong.
Description: Cambridge, Massachusetts : The MIT Press, [2024] | Includes bibliographical references and index.
Identifiers: LCCN 2023027971 (print) | LCCN 2023027972 (ebook) | ISBN 9780262548052 (paperback) | ISBN 9780262378659 (epub) | ISBN 9780262378642 (pdf)
Subjects: LCSH: Computer science—Study and teaching—Case studies. | Problem solving—Study and teaching—Case studies.
Classification: LCC QA76.27 .C478 2024 (print) | LCC QA76.27 (ebook) | DDC 004.071—dc23/eng/20230905
LC record available at https://lccn.loc.gov/2023027971
LC ebook record available at https://lccn.loc.gov/2023027972

ISBN: 978-0-262-54805-2