Research Questions and Approaches for Computational Thinking Curricula Design

Eleni Berki¹, Juri Valtanen², Aleksi Tiensuu¹ and Kira Grigoriadou³

¹Faculty of Natural Sciences and ²Faculty of Education, Kanslerinrinne 1, Pinni B/Virta Building, University of Tampere, 33014, Finland eleni.berki@uta.fi; valtanenjuripetri@gmail.com; aleksi.tiensuu@gmail.com;

³Science and Engineering, Faculty of Natural Sciences, Tampere University of Technology, PO Box 527, 33720 Tampere, Finland kira.grigoriadou@student.tut.fi

Abstract

Teaching computational thinking (CT) is argued to be necessary but also admitted to be a very challenging task. The reasons for this, are: i) no general agreement on what computational thinking is; ii) no clear idea nor evidential support on how to teach CT in an effective way. Hence, there is a need to develop a common approach and a shared understanding of the scope of computational thinking and of effective means of teaching CT. Thus, the consequent ambition is to utilize the preliminary and further research outcomes on CT for the education of the prospective teachers of secondary, further and higher/adult education curricula. This research study comprises a proposal for carrying out research and development practices regarding the teaching and integration of mathematical and computational thinking in the curricula. The emphasis is put on the following research agenda, aiming at: a) clarifying the meaning of CT and its scope, b) identifying cross-curricula and other approaches for teaching CT, c) providing a generic curriculum design for CT for students and teachers and d) finding possible obstacles, limitations and bottlenecks for the realisation and evaluation of the previous.

Keywords: Computational Thinking (CT); Mathematical Thinking (MT); Curriculum Design (CD); Further Education (FE); Higher Education (HE).

1.0 Background and Introduction

An interesting question that recently occupies the minds of computer science researchers, teachers and curricula designers is:
Why computational thinking is on the teaching focus for achieving/advancing digital literacy?

It is said that there is no better way to answer a question than asking another question. Thus, herein we attempt an answer, by posing many other questions!

What is computational thinking and how is it contextualized?

There seems to be an obvious genuine need for a robust and agreed definition and a scope of computational thinking (CT), in order to identify its teaching aims, purposes and learning outcomes better. Since Wing in 2006 [1] popularised the term ‘computational thinking’, which was coined by Papert in 1980 [2] and 1996 [3], there has been confusion about what exactly computational thinking means; Papert associates clearly CT to generative and generic mathematical thinking and associated mathematical education. Jones in 2011 [4] argues that Wing [1] never really gave a solid definition of what exactly computational thinking is; only a number of characteristics of computational thinking. Perhaps for this reason Wing’s understanding of computational thinking encompasses a very wide range of ideas; everything from ‘a way that humans, not computers, think’ to ‘for everyone, everywhere’ (p. 35). Yet, humans think in a very large variety of ways [4].

Is there a need to re-define, refine and deepen the concept and scope of CT?

Some even have asserted that there is no need for a precise definition of CT. For example, Guzdial in 2011 [5] supported the idea that a very broad definition is enough. However, this runs the risk that the focus will be moved from what CT is to how it should be taught. The obvious importance of digital literacy in our era can be evidenced from the attention it is paid to, ranging from primary school to job careers in the adulthood. People’s every day activities from primary school to adult education and future careers have been fully automated or technology-based that many require from the ordinary user to have some knowledge of computer technology. Thus, programming skills, computational modelling skills and associated thinking skills have become a mandatory part of everyday routine to an extent that school curricula should cater to provide these for future citizens. Computing, analyzing, representing or selecting data nowadays happens with the help of computer-oriented actions. No one can deny the importance of programming and computation in everyday life, and this is the reason and the need to introduce relevant knowledge and make programming and CT a part of the curriculum at schools. Notwithstanding, the challenging questions still remain:

How to teach programming and CT and who could teach them best?

Admittedly, using software and hardware, which are vital parts of computer programs, is not so difficult. Pressing some suitable buttons and getting some useful results can be taught during a short time. But how can someone make that machine program do exactly what is precisely wanted to do in a correct way? That is the question, which brings to the deliberate action for becoming a teacher of
programming and computational modelling at a more abstract level. For being able to make the mysterious machine to do what you want, you need to find an acceptable and provable way to communicate with it, ideally as you would have to do if you needed to communicate with another human. But there is a problem - this machine cannot understand you (and your natural language) as cannot understand many other people worldwide (and their different languages). For this reason, programming languages have been invented. Each one serves a specific purpose and makes programmers’ lives ‘somehow’ easier, or at least, less complicated.

Learning a programming language for being able to do the basic computing operations is not so difficult, as it is not so difficult to learn any human language for covering the basic functions in everyday life. But learning a spoken language for understanding the meaning between the lines in conversation or in written text, learning to understand the jokes or generally “feeling” the language, is not an easy task. This has many requirements and many conditions that affect the learning process. Generalizing this idea in terms of a suitable programming language, someone can observe that it is not so hard to learn the basic commands of one specific programming language, or it is not even hard to write some simple lines of code; but coding general computable solutions for any given problem is a challenging task and also has many pre-requisites and needs. Two of them, and probably the most important, are to be able to a) think and b) communicate with the mysterious computing “machine”. This, for the record, could only and in a very broad sense, be called computational thinking (CT)!

In mathematics teaching culture the latter sounds like the norm on how CT should be explained to students. For a mathematician though, it is necessary to define the concept from an abstract point of view before analysing it. Apparently, the need for teaching CT is not new; it has only recently been emphasized and popularized [6]. In Finland (and many other EU countries) [6] the new curriculum makes it obligatory to teach some basic ideas of programming and computing independently or during the maths lessons in primary, middle and high school. Further, since the final examinations of the Finnish high school (OY kirjoitukset) are becoming computer-based from 2019, the need for an increased digital literacy is obvious.

2.0 Talking About and Beyond Computational Thinking and Computational Modelling

Apparently there seems to be no ‘current expert’ to define the apparent need and relation of computational thinking and digital literacy, but tentatively speaking the digital literacy’s preliminary role seems to be somewhat parallel with computational thinking. It seems to be like a subset of the skills used by a computational thinker and modeller and at the same time, depending on the definition used, reaching the borders of computational thinking. Earlier research in computational modelling (see e.g. [7, 8, 9, 10, 11, 12, 13, 14]) has touched the need for CT by branching to the use of quite specific thinking skills for designing architectures for services and devices, associating CT and modelling to art, science, education, social interaction, problem-solving, software development tools,
systems design and so on. However, as the use of devices extends the mental
capabilities of the problem-solver, accessing the vast interpersonal domain of
knowledge to extend the personalised knowledge of the problem-solver, and
interacting with the other people to generate solutions, both have an important role
in modern day problem-solving processes. The latter of course depends on the
problem at hand, and in clear computational thinking there are parallel dimensions.

Is there a need for educational approaches to develop computational thinking
skills for complementing problem solving skills?

Problem-solving does not begin when a problem is first encountered. The process
starts way before that. By and large, the concepts, methods, models, and
knowledge (e.g., scientific results) that we use in problem-solving are not
generated by an individual, but derived from the surrounding world. Learning is
inextricably linked to problem-solving. The process of internalizing the external
knowledge, training, and experiences, including e.g. strong (domain-specific) and
weak (more domain-general) problem-solving methods into cognitive and
metacognitive skills/knowledge, and further refining and connecting the mental
schemata is a life-long (learning) process.

It is known that cognitive skill does not transfer well, meaning that something
useful from one domain might not (subconsciously) get triggered in another. Also,
when a method from the interpersonal domain is internalized into a mental
representation of it, it is likely to lose at least some of its potential abstract
usefulness. Even when a weak problem-solving method is presented in abstract
fashion there still is a context for it to bind to which is the abstract presentation
itself. These are important to keep in mind when considering a good way to teach
computational thinking to people. Therefore, not only what to teach in CT, but also
how to teach CT are very relevant and valid considerations.

In relation to thinking and problem-solving, CT can be defined around two distinct
dimensions. Naturally, it is a set of mental tools and strategies that can be
consciously applied in problem-solving situations. However, one might, again, ask:

Will the mental tools and strategies of computational thinking be applied?

Does the possibility of application occurs in the individual learners’ conscious
minds?

It is understandable to take the above questions on board since these issues have a
lot to do with the subconscious cognitive processes. These tools and strategies exist
in the domain of interpersonal knowledge, and they are important parameters in the
evaluation of the effectiveness of the teaching of CT.
2.1 Intrapersonal Computational Thinking

Computational thinking also exists intrapersonally; it is something that starts forming in an individual’s mind in the process of internalization, where the mental representations of the interpersonal domain knowledge are formed.

One of the key questions to address, after the content of computational thinking is defined on the interpersonal level, is:

*Should the approach to teach computational thinking be from abstract to specific or from specific towards abstraction?*

The above question relates to the exemplar and prototype theories. There is much argumentation and controversy on the above discourse, sometimes to pose another (about relevant scope and purpose) question:

*Should CT teaching and learning be tackled from both directions?*

If so, how could one do it without increasing the cognitive load too much? This also relates to the role of use of the episodic and semantic memory (that are neurologically distinct systems) in the learning process. Thus, the previous two questions could be re-shaped as follows:

*Is it better to learn by experiencing and abstracting from the experiences, by combining with experiences in different domains, or try to learn more in relation to semantic memory?*

2.2 What Distinguishes a Computational Thinker from Other Thinkers?

Working with well-structured concepts and environments is central in computing and computational science; thus the computational thinker is familiar with dealing with these types of problem-environments. These environments are also known as high validity environments. While interacting within high validity environments it is possible for a person to develop something called expert intuition, which means that the subconscious processes are able to generate a valuable hypothesis (i.e. good creative ideas) and make them available for the conscious mind. These kinds of explicit processes play a meaningful role in human problem-solving.

2.3 Computational Thinking applied in Problem-Solving

When computational thinking is applied in more ill-structured problem-environments, it is important to be aware of it, since in ill-structured environments things work differently. Different kinds of approaches are required and the intuitions can become unreliable. (With ill-structured concepts and environments expert intuition is unlikely to develop, because it requires somewhat clear and immediate feedback of the correctness of the provided result, which in generally is
not available when dealing with ill-structured problems). This is something to be paid attention to, when the cognitive skill transfer is tried to be facilitated by applying computational thinking in different fields. There are also other possible biases that computational thinking can cause when applied in other domains. These should be mapped out in order to figure out a good approach to guide to transfer CT between domains. Compared to other types of sets of mental tools, e.g. De Bono's six thinking hats [15], one can observe that the mental tools provided by CT have a clear emphasis on targeting to guide to interact with interpersonal things (concepts, processes, phenomena, etc.) and not so much with the intrapersonal ones.

3.0 On suitable Research Questions and Research Methodologies to advance knowledge on CT

Following the results of the preliminary literature review of Tiensuu in 2012 [16] and 2013 [17], we have reached the conclusion that there is a need to proceed to the next three metalevel generic and strategic research questions:

**RQ1: What research is needed to further the CT agenda in formal and non-formal education settings?**

- Whatever research aim/activity should be able to show clear cognitive benefits of computational thinking in comparison with other types of thinking; which is research that has not been done. The outcomes should also show that there are meaningful transfers that can happen in reality; which, in turn, would be a good thing, as some of these weak problem-solving methods are criticised for being too general to actually offer any help with anything a little more challenging.

**RQ2: Can/should schools provide all school-children with learning experiences that aim to nurture their CT skills (in all school subjects)?**

- Maybe, but probably mostly within other topics as domain specific problem-solving methods are superior, compared to domain general ones. This would be a safe approach through suitable problem choice and problem-based learning. Even if this new type of thinking does not produce many benefits, at least school-children and (future) work and society still get their strong methods for problem-solving in diverse contexts.

**RQ3: Should mathematics teachers teach computational thinking to the students?**

- This question often comes up due to the shortage of resources (e.g. staff and time). Mathematics teaching is in the curriculum for making the students familiar with mathematical concepts, teach them to solve problems, but also to think in mathematical (also computational) ways, which is a demanding
task. It might even be a good idea not to have an interdisciplinary or multidisciplinary approach at the initial stages of learning how to think mathematically and computationally. For instance, when the target group is middle school students, who do not have yet a solid knowledge of what is mathematics and what is computer science, the teaching and learning methods can vary and can be applied differently for the exposure of similar or different concepts. Further research and development is needed to find out.

Subject specialists could probably embrace creative methods of teaching and learning providing that they want to transfer their specific knowledge and subject passion to others. For this reason mathematicians might be best to teach mathematics at school, in the same way as programmers should teach programming and computer scientists should teach computational thinking. After all, subjects specialists overcome the general and basic knowledge provided to teach those subjects in the school level, and have the passion for being subject-specific in many creative teaching ways. Their teaching and learning methods can vary a lot and can be applied differently for the exposure of different concepts. Lozanov’s teaching methodology [18], for instance, is a student-oriented methodology, but it can change from one teacher to another, from one subject application to another. The teacher, for instance, can design the method application of making the students familiar with mathematical thinking, but can (and should) also design another application for making them familiar with CT.

4.0 Computational Thinking v Mathematical Thinking: Similarities and Differences

Apparently, there are many similarities between mathematical and computational thinking, but by no means can someone claim that they are exactly the same. MT is broader while someone should consider CT as a subset of mathematical thinking. If someone thinks about computational thinking using the approach of Tiensuu in 2012 [16], where computational thinking is part of problem solving, it can then be asserted that CT is a subset of mathematical thinking. However, mathematics is not (only) the science of solving problems, but (as many people joke about it), it rather is a science of creating a problem and then solving it.

Adopting the second approach of Tiensuu in 2012 [16], where computational thinking is a subset of computer science (mathematicians might agree more with that definition-approach), we can see more similarities with mathematical thinking. In both mathematical and computational thinking, there are also precise definitions and logical deductions. The logical deduction is rather a stronger concept in terms of mathematical thinking but it can still exist in pure computational thinking.

Moreover, in both ways of thinking, the ideas of classifications, conjunctures and organisation are the same. For example, in mathematics we set our domain of definition in the beginning and in programming there is a need always to become familiar with the basic operators and commands of each language. Both types of
thinking need organization - in mathematics it is called axiomatic (logic); but the same idea applies to CT as well. Finally, we can claim that there is a certain pattern in both types of thinking, comprising the three following constructs: i) the beginning of it - dissembling what we have in our hands, ii) the main part - the solution of it, and iii) the conclusion. Through these constructs, one can see that MT and CT include critical and creative thinking in their processes, i.e. in problem finding.

Often in their thinking mathematicians are somehow (but not entirely) different from computer scientists. Comparing and contrasting MT and CT, we can see that mathematical thinking has inside of it a beautiful and harmonic way of seeing the surroundings, something that probably computational thinking does not have in that extent, or it is not so obvious. Fibonacci sequences, for instance, can be viewed in musical scale and the result can be a wonderful piece of art; which could be viewed in a rather deterministic way through the lenses of CT.

Furthermore, in mathematics you can find the natural idea of abstract thinking, which cannot be found anywhere else in the same sense. Even computational structures possess abstraction in a concretized and certain way. In high level mathematics the interpretations of everyday life can be smashed and become universal. Five plus one is not always equal to six (5+1=6?) but can be, for example, equal to two. High level mathematics provides the students with the chance to see the whole world through a prism, where they can observe different wavelengths and can see the white light having many colours inside. They can start associating this stage of thinking with philosophy and could understand the idea of cosmos. They can learn to accept different people, their different ideas, become more tolerant and understand the beauty of diversity. They will learn that there is not only one correct answer, because the answer depends on which is the current domain and on which topological space we are finding and solving the problem.

The beauty of abstract and versatile mathematical thinking cannot be found and experienced in the scope of CT, since the one is a subset of the other. Accordingly, the teaching and learning methods can vary a lot and can be applied differently for the exposure to MT and CT. Lozanov’s teaching methodology [18], for instance, is a learning-centred method, versatile in its application to subject-specific knowledge. The method itself is being praised worldwide for having a significant effect for teaching languages at a large extent, but also science and mathematics successfully. Subsequently, it would be interesting to see the effectiveness of known teaching methods in the teaching of programming languages, mathematical or computational thinking.

5.0 Computational Thinking in Curriculum Design

Immediate benefits that could be obvious from a further systematic literature review with well-defined inclusion/exclusion criteria and collection of experts/policy-makers opinions, are:

a) a proposed definition/clarification of computational thinking and its scope, and
b) a suggested curriculum design for teaching CT with clarity and effectiveness.

Yet, there are many less consistently used terms while attempting to define conceptual thinking. These non-consensus terms can be classified into four areas:

i) Thinking terms; there are suggestions that several specific types of thinking (logical, algorithmic, engineering and mathematical) should be included. Yet, of all the potential terms associated with thinking, only algorithmic thinking is the possible term which may be suitable for inclusion in a definition for computational thinking.

ii) Problem solving terms; the idea that CT has some relationship to problem solving appears frequently in literature. The most frequently employed specific terms in discussions of general problem-solving skills are problem-solving, analysis and generalisation. Yet, problem-solving, while consistently used in literature, is a broad but not well-defined term. Analysis, used in the context of solution, is analogous to evaluation and used consistently. The term generalisation is used infrequently but there are descriptions of analogous processes. For this reason, the suitable terms for inclusion in a definition of computational thinking are evaluation and generalisation.

iii) Computer science terms; CT has a deep relationship with computer science and some specific terminology has suggested to be included in a definition, such as systems design and automation as well as more general terms such as recursion and recovery through redundancy. Yet, none of them appears suitable to be included in a definition of computational thinking, since systems design and automation is evidence of the use of computational thinking skills, not a definition of it. Moreover, those terms that are interpretable as computer science content do not bring focus to the definition of computational thinking.

iv) Imitation terms; terms modelling, simulation and visualisation appear frequently in literature. Yet, it is the manipulation of abstractions (models, simulations, and visualisations) that contribute to the development of computational thinking skills, but do not necessarily define it. That is, these tools are effective aids in developing computational thinking skills, but they may not be suitable for inclusion in a definition of computational thinking.

For this reason, Selby and Woollard in 2014 [19] gave the proposed definition of computational thinking that ‘computational thinking is an activity, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects the ability to think a) in abstractions; that is a process of making an artefact more understandable through reducing the unnecessary detail, b) in terms of decomposition; that is a way of thinking about artefacts in
terms of their component part which makes complex problems easier to solve, c) algorithmically; that is a way of getting to a solution through a clear definition of the steps by thinking in terms of sequences and rules, and reach a solution that works every time, d) in terms of evaluations; that is a process of ensuring that a solution for example an algorithm fits for purpose, and e) in generalisations; that is identifying patterns, similarities and connections, and exploiting those features quickly solving new problems based on previous solutions to problems.’

That is, CT is a focused approach to problem solving, incorporating thought processes that utilise abstraction, decomposition, algorithmic design, evaluation, and generalisations. Selby and Woollard (2014) [19] acknowledge that the definition can change as understanding of CT develops over the coming years.

### 6.0 Troubles with CT that Experts and Policy-makers Must Resolve

In addition to the exact definition of CT, Jones (2011) [4] argues that Wing (2006) [1] never really explained how computational thinking differs from other kinds of thinking that require abstraction or a great deal of data. In addition, Jones (2011) [4] finds it difficult to accept that CT can be used to solve every problem as Wing [1] seems to inform us. Instead, Jones argues that there are a number of problems that would be seen to lie outside the realm of CT such as questions of aesthetics. Also, according to Jones (2011) [4] it seems that it would be difficult to solve something like a moral problem or a question of ethics based on the system of CT.

Since such problems are not a process of amassing data and drawing conclusions based on that, or even a process of conceptualising, rather they are a process of understanding personal values and ideas, or values that belong to a society as a whole. Moreover, according to Jones (2011) [4] and many other pedagogists, computational modellers and practitioners (see e.g. [12, 13, 15, 16, 17, 20]) there are problems that do not always have a concrete and reachable solution. After all, CT has limits and limitations that are in the nature of thinking itself. An effective, clear, consistent and relevant problem-based curriculum for CT should cater for all these challenges and bottlenecks.

Last, the authors believe that computational thinking and modelling, as subsets of mathematical thinking and modelling [20], could and should be combined with generative, parallel and manifold thinking [21] and should also be observed in its application in both informal and formal learning [22, 23] settings.

### 7.0 A Thematic Framework for Interviewing Experts and Policy-makers

A thematic framework of semi-structured interviews of computational thinking experts is unfolded below.
1. Understanding of computational thinking (CT) and its teaching

- What is computational thinking?

- How CT is similar and/or different with coding, programming etc.

- To whom computational thinking and its teaching is important? Why?

2. Attitude towards computational thinking and its teaching

- How the field of computer science feels about CT and its teaching in general?

- How other fields of science feel about CT and its teaching in general?

- How do you personally feel about computational thinking and its teaching?

3. Experiences on computational thinking and its teaching

- What kind of experiences the field of computer science has about CT and its teaching?

- What kind of experiences you have about CT and its teaching?

4. Lessons learned from computational thinking and its teaching

- What kind of lessons the field of computer science has learned about CT and its teaching?

- What kind of lessons have you learned about CT and its teaching?

5. Vision(s) of computational thinking and its teaching

- How do you think the field of computer science sees the near and far future of CT and its teaching?

- How do you personally see the near and far future of CT and its teaching?

Influential policy-makers and international specialists on CT and related areas need to be contacted for questioning and/or interviewing for: i) accessing implications for educational policies and practices; ii) commenting on bottlenecks and barriers encountered in the implementation/evaluation of CT; iii) providing background information on the up-scaling and in the realisation of CT; iv) streamlining in-
depth knowledge by subject professionalism; v) outlining experiences from involvement in CT framework initiatives for education.

8.0 Summary, Conclusions and Future Research

Summarising, our preliminary literature review on CT pointed to further questions and considerations that should be taken into account for future research. According to Selby and Woollard (2014), based on literature review focusing on the term’s consistency of use and interpretation, there appear to be three consensus terms that a definition of computational thinking should include: the idea of i) a thought process and the concepts of ii) abstraction and iii) decomposition.

The practical use of CT and computational modelling skills as part of digital competencies is an area where European education schemes seem to have a significant competitive edge. There has been much current emphasis on teaching CT from the point of view of educational research and curricula design. Henceforth, suitable pedagogical methods for teaching CT as part of mathematical thinking are in search for accelerating learning. There should be a future better, closer view of the results of literature reviews and the involvement of specialists and practitioners on CT for additional questioning and interviewing.

The questions we considered here tried to sketch the CT (skills and knowledge) domain and capture the essence of CT. If there are (also) experts (as we assume there could and probably even should be) that are not familiar with CT, they cannot probably answer to this kind of interview questions as we/they assume some degree of knowledge about CT and/or computational modelling. Such experts can be, for instance, human cognition or programming gurus, whose opinions can be very valuable for reaching possible outcomes and policies.

The potential experts to be used would not probably be able to give their reliable views and some potentially valuable input about a relatively new old concept such as CT, unless it is first defined for them. On the other hand, not been earlier acquainted with the exact and consistent definition(s) of something, one can actually and probably focus on something close but different, that is somewhat familiar with; and this prevents from addressing the essential research context.

7.0 References


17 Tiensuu A. A View: Computational Thinking in Regard to Thinking and Problem-Solving, proceedings of INSPIRE 18, pp83-92, 2013.


