The uniqueness of Computational thinking

P. Larsson*, M-V. Apiola* and M-J. Laakso*

* University of Turku/Department of Future Technologies, Turku, Finland
peter.larsson@utu.fi, mikko.apiola@utu.fi, milaak@utu.fi

Abstract - The concept of computational thinking is timely. It is widely agreed that in order to benefit and thrive in a digital world, more than digital literacy is required. Computational thinking promises understanding of information technology and its utilization without necessarily knowing how to write code. According to many authors, skills in computational thinking include the ability to evaluate and design solutions to problems. However, the vision for computational thinking, laid out by a number of scholars, is even bigger than this. It proposes that thinking habits from computer science are beneficial for any kind of problem solving and that those thinking habits form a central part of modern sciences. It might be that components of computational thinking are indeed useful in problem solving and that they play an important part in other sciences. However, computer science combines three types of thinking traditions: theoretical (mathematical) tradition, engineering tradition and scientific tradition, and it is important to differentiate between what is an aspect in one or more of these traditions and what is unique for computer science. This clarification will support the application and teaching of computational thinking. This paper explores the uniqueness of computational thinking and its roots in theoretical, engineering and scientific traditions of computing.

Keywords - Computational thinking; Computer science education; CSER; Computational ideas; History of computational thinking; Disciplinary ways of thinking and practicing

I. INTRODUCTION

The ongoing digitalization of most functions of society has followed the ubiquity of computing technology. Digitalization is not anymore about technology, but how we do things and what kind of form our organizations are taking. The change brings new opportunities and benefits, but also possible negative effects. It has been estimated that 47% of US jobs [1], and one third of Finnish jobs [2] are in risk of being automated in the next decade or two. Even though these estimations might be exaggerations, it is likely that automation will be a part of most of our occupations in the future [3]. Those who have the necessary skills will be able to benefit from this change [4]. In such digital world Computational thinking (CT) is a foundational competency for being an informed citizen [5].

The school should prepare pupils for the new digital world. Along with the digitalization of our society, also our schools are getting digitized. Especially in STEM subjects, CT skills are central [5]. This doesn’t necessarily mean that CT always involves use of computers. CT is a way of thinking that has developed in Computer science (CS). Jeannette Wing [6] located CT in the intersection between mathematical thinking, engineering thinking, and scientific thinking. Engineering, mathematics and science also constitute the basic pillars of CS [7]. CT is a movement, which has only recently started to look for its identity. This means that there is no clear consensus as to what CT is, and no clear guidelines on how to teach CT for different learner groups. Denning [8] noted that teachers still ask, “What is computational thinking?” This paper’s aim is to contribute to finding an answer to that question.

The article is structured as follows: in the following section, related research is presented. The third section discuss issues with the definition of CT and recognize that there is need to clarify how it is interpreted. The theoretical, engineering and scientific ways of thinking provide perspectives on CT, but there is also a need to identify what makes it unique. The fourth section provide a framework for describing the perspectives using seminal examples. In the fifth section the examples are analyzed for components of CT to illustrate the differences of the perspectives and find the uniqueness in CT. The final section present’s the conclusion of the paper.

II. RELATED WORK

CS is multidisciplinary and continually evolving research field. Discoveries are transforming existing research and new subfields are born. The field of CS can now be classified into 18 knowledge areas. This complexity makes it difficult to agree on a definition of CS [9]. The history of the field let us examine its core traits at a time when the complexity was more manageable. This work relates to uses of the historical perspective to analyze CT for teaching or learning contexts. Matti Tedre and Nella Moisseeinen [10] investigated the notion of CS as an empirical science by going through examples and opinions of experimental research from decades of the field’s evolution. Since CS field is multidisciplinary, the terminology used can be conflicting. Their aim was to clarify the terminology and contribute to the disciplines self-understanding through the examples. Matti Tedre and Peter Denning [11] looked at how thinking and practicing in CS, CS education and its research, and computational science and society’s digitalization has evolved. Their aim was to support the development of CT by warning of past mistakes and informing of great ideas. They also identified different threats to current CT initiatives. Benvenuti et al. [12] took an historical perspective when offering guidelines for the definition of the computing part of hybrid computing curriculum. They recommended following Tedre’s and
Apiola’s [7] observations that educators should understand the complex nature of computing by being familiar with how theoretical, engineering and scientific styles of computing culture manifests in the different topics. If some style is neglected, the student’s lack of theoretical, technical or scientific understanding will put them at disadvantage compared to a more balanced teaching.

III. ISSUES IN THE DEFINITION OF CT

CT promises abilities to solve problems, design systems and understanding human behavior using concepts fundamental to CS. Jeannette Wing made this idea popular in her article in the Communication of the ACM [13]. She claimed that everybody would benefit from learning the thinking habits of computer scientists. Wing showed examples of useful concepts from different parts of the CS field but didn’t offer an explicit definition of CT. CS is a very broad field [9] and this opens for variety of possible interpretations. Shute et al. [14] literature review shows a diversity of definitions, interventions, assessments and models of CT in education. There is a lack of generally agreed-upon definition of CT and its main components. This makes it difficult for teachers to incorporate CT in their teaching [14].

Seymour Papert [15] coined the term CT meaning a method where programming is used for learning mathematics. This was self-guided learning where domain specific functions and feedback from the computer gave support to the student. Later Jeanette Wing [13] reintroduced CT and defined it as thinking patterns used by computer scientists that entailed more than programming. In the context of the Scratch-programming environment, CT has been about computational concepts, practices and perspectives [16]. Children engage with programming concepts, form practices while using the concepts and through their engagement gain perspective of the world and themselves. Peter Denning [8] quoting Aho [17] defined CT to be about defining algorithms using a computational model. In a recent definition based on cognitive science computation is replaced with modelling and simulation [18].

There seems to be differences in interpretation of CT even within a small sample of definitions (above). Shute et al. [14] set out to demystify [19] CT by giving a definition and creating a model of the core components based on a literature review. They [14] defined 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”. The core components were defined as decomposition, abstraction, algorithm design, debugging, iteration and generalization.

We find the definition made by Shute et al. [14] of CT and its components well founded and welcome clarification. However, with the goal of making the definition general also the gist of CS seems to be lost (except the mention of computer). Since the intended audience (teachers) might be from outside of the field, they might lack the necessary intuition of what CS is. CS combines theoretical, engineering and scientific perspectives [7]. This can make it difficult to separate what might already be a part of a subject [20] and what CS can contribute. In this paper, the aim is to clarify how the perspectives differ and what is unique to computational thinking by using a framework of the three CS perspectives.

IV. A FRAMEWORK DESCRIBING COMPUTATIONAL THINKING PERSPECTIVES

Computational thinking consists of theoretical, engineering and scientific perspectives [6]. Teachers should explicitly consider what emphasis is placed on the different perspectives in teaching CS or utilizing computational thinking in other subjects [12]. If the teacher doesn’t have a background in CS, it might be difficult to separate the different perspectives and it might also be difficult to spot what is uniquely Computational thinking. We share the historical approach of Benvenuti et al. [12] and use the task force on the core of CS report [21] as a framework to answer these questions. The report describes theoretical, engineering and scientific perspectives as distinct problem-solving processes consisting of four steps. Here we describe the steps of the processes and motivate the choice of a seminal examples for further illumination.

The theoretical perspective has its roots in mathematics where the central issue is theorem proving. First, the objects of study must be described using rules to define their properties and relationships (axioms). Then a possible new relationship between the objects can be suggested (theorem). The theorem is proved by deduction where axioms and other theorems are combined using the rules of logic. The result is interpreted to motivate why it is significant or useful. Of all the models of computation, the Turing machine [22] is regarded as the one representing a theory of CS. We use the original proof that the decision problem is unsolvable as an example of the theoretical perspective.

The engineering perspective of CS began with Electrical Engineering, which later developed into Computer Engineering when the basic logical computer components had been invented [23]. The aim of engineering is to solve a problem by constructing a system or device. First, the requirements related to the expected solution of the problem is identified. Requirements are turned into exact specification, which governs the development of the system. Following the specification, the system is designed and implemented. The system is tested for conformance to the specification. ENIAC – The Electronic Numerical Integrator and Computer [24] is generally held as the world’s first computer. It is chosen as an example of the engineering perspective since it can be seen to exemplify pure engineering before the von Neumann architecture introduced mathematical logic to be a part of designing computers.

Of the three perspectives, the scientific, which refers to empirical inquiry, came last. Its aim is to explore some phenomena by devising experiments. First, a hypothesis is formed about some real-world phenomena. Then a model is constructed which allows to make predictions of its effects. To test the model an experiment is designed, and
data is collected from its execution. The results are analyzed and compared with the predictions. Allen Newell and Herbert Simon [25] stated The Physical Symbol System hypothesis in the context of artificial intelligence. The hypothesis is chosen as an example since it was presented in a Turing award lecture in 1975 where CS was interpreted as an empirical science. This was the last of the perspectives and adjoining examples, which are summarized in TABLE I.

TABLE I. COMPUTER SCIENCE PERSPECTIVES AND ILLUSTRATIVE EXAMPLES

<table>
<thead>
<tr>
<th>Step</th>
<th>Theoretical</th>
<th>Engineering</th>
<th>Scientific</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Define objects of study</td>
<td>State requirements</td>
<td>Form a hypothesis</td>
</tr>
<tr>
<td>2</td>
<td>Hypothesize possible relations between objects</td>
<td>Define specification</td>
<td>Construct a model and make a prediction</td>
</tr>
<tr>
<td>3</td>
<td>Determine if the relationship is true</td>
<td>Design and implement the system</td>
<td>Design an experiment and collect data</td>
</tr>
<tr>
<td>4</td>
<td>Interpret the results</td>
<td>Test the system</td>
<td>Analyze results</td>
</tr>
<tr>
<td>e.g.</td>
<td>Turing machine</td>
<td>ENIAC—the world’s first computer</td>
<td>The Physical Symbol System hypothesis</td>
</tr>
</tbody>
</table>

The previous three examples will be used to describe the problem-solving processes of the theoretical, engineering and scientific perspectives of CS. We will later use the Shute et al. [14] components of Computational thinking to highlight the differences of the perspectives. At the same time, the examples are searched for a commonality that could said to be the gist of Computational thinking.

A. The Theoretical perspective – Case: Turing machine

Alan Turing [22] proved the Entscheidungsproblem unsolvable and was going to publish the result in an article 1936, but Alonzo Church [26] beat him to it by having published an equivalent result earlier. Both articles made an explicit definition of an algorithm and using that definition showed that an algorithm that could decide if a formula is provable is impossible. Even though the results and the method are similar, the definitions of an algorithm were different. Church created a mathematical formalism, lambda calculus, using the concept of a function. Turing mechanized the act of calculating until it could be described as a process performed by an imaginary machine, Turing machine, governed by rules. To calculate with Lambda calculus, you have to understand how to use the terms, but the Turing machine and the rules controlling it didn’t require understanding anything more than was shown. As a definition of an algorithm, the Turing machine was deemed more explicit [27].

The creation of the Turing machine can be presented with the four steps of the theoretical perspective. Entscheidungsproblem or the decision problem was stated by David Hilbert as the question of the possibility of a procedure, in today’s parlance algorithm, that would allow mechanically, without any thought (according to talk by Behman 1921 in [28]), to decide if a well-formed formula is provable. The objects of study are algorithms. Given a formula, the result of executing the algorithm is true if the formula is provable and false if not. Every step in the decision algorithm has to be explicit to qualify as a proof and therefore we need an exact way to define an algorithm.

Turing simplified human calculating on a paper to create an explicit definition of an algorithm. The paper can be replaced with an infinite tape divided into squares. Only one square is under work at any time and one symbol can be written per square. The symbol written (or erased) and the movement on the tape are governed by rules. To simulate the state of mind of the calculator every rule point to the next rule. There are different versions of rules depending on the symbol that is read from the active square. Turing’s definition of an algorithm was an imaginary machine that a human could use to perform a computation mechanically following the rules without any understanding of what was calculated. The rules of one machine could perform a particular computation. To make the proof Turing needed machine that could perform the same calculations as another machine. He made a universal machine so that it could emulate another machine if its description was given on the tape.

Turing suspected that the decision algorithm was not possible. First, he showed that it is impossible to devise a machine that could in general, utilizing a finite process, calculate if another machine would print the digit zero. Then he restated the description of an example machine presented in the paper using the language of first-order logic. He made a proof-machine that took another machine’s logical description as an input. The proof-machine executed a logical formula which stated that the machine given as input was only provable if the machine ever printed zero. Turing proved the formulas correctness and since he had earlier shown that, this kind of machine is impossible it showed that the decision problem is not solvable.

B. The Engineering perspective – Case: The ENIAC computer

ENIAC – The Electronic Numerical Integrator and Computer [24] often regarded as the world’s first electronic computer was built originally to calculate ballistic trajectories for different kind of artillery during the World War II. In 1943 in preparation to join, the war in Europe USA accelerated the creation of new types of artillery and shells. Each combination needed firing tables to aid the use of the gun in field conditions were there were no time to perform complex calculations. The firing tables were calculated in the army’s Ballistic Research Laboratory in Aberdeen Proving ground in Maryland using mechanical desktop calculators and a differential analyzer. Even with mechanical aids, the calculation of one firing table took three months [29]. The development of ENIAC was proposed as a solution in April 1943.

The creation of ENIAC can be described with the four steps of the Engineering perspective. Preparing firing-
In April 1943 John William Mauchly, J. Presper Eckert and John Brainerd submitted a proposal for the ENIAC to the army [30]. The ENIAC was presented as an all-electronic version of the mechanical differential analyzer, which would perform the same calculations with greater speed (100 to 1000 times faster) and accuracy (mechanical analyzer was not as accurate as a human computer). The machine structures were similar; the original differential analyzer had several integrators that did the calculations and connecting rods transferred the values where ENIAC had accumulators and connecting wires. However, the operation principle was different since in ENIAC the accumulators held the values and the wires represented functions. The control of the operation in ENIAC was modeled after how human computers arranged their calculations. The idea was new and its implementation an open question. The submitted design of ENIAC was a combination of the two models of computation [30].

The work on the ENIAC was an engineering project which aimed to realize the design presented in the proposal, not to pursue further innovations [30]. ENIAC consisted of nine different type of units which all had to be designed, tested and debugged [31]. Central to the ENIAC architecture where twenty accumulators and the master programmer [32]. When an accumulator receives a number, the number is added to its store. Each accumulator was locally programmable through switches to control how many times numbers were received and results transmitted. When an accumulator had finished operation, it transmitted a control signal to activate the next unit or units. How the control was transferred were done through wiring for instance to add a number one accumulator triggered another two to transmit and receive a number. The operations of the units were synchronized through signals from a cycling unit. The master programmer controlled the operation of other units allowing for repetition of a set of operational steps.

C. The Scientific perspective – Case: Physical Symbol System

Allen Newell and Herbert Simon held their ACM Turing Award lecture in 1975 entitled “Computer Science as Empirical Inquiry: Symbols and Search” [25]. They compared their topic with previous award lectures and noted that their presentation brought a new perspective on CS as an empirical inquiry. Experiments are central to CS, but they differ from how experiments are traditionally understood. In CS it is the execution of the created combination of hardware and software that makes the experiment. Even though the method is unique, the goal is the same than in other sciences to discover new phenomena and analyze ones already known. Newell and Simon used symbol systems and heuristic search from their research on artificial intelligence to exemplify empirical research. We will use the investigation of the Physical symbol system hypothesis to illustrate the process of the scientific perspective.

The Physical Symbol System Hypothesis states: “A physical symbol system has the necessary and sufficient means for general intelligent action.” [25, page 116]. By being physical the system is tangible and obeys the laws of physics. According to Newell and Simon [25] such a system can also be engineered from a set of components. A symbol system consists of symbols that form expressions which are structures where symbols are related by some organizing principle. The system is not static, it contains processes that operate on expressions to produce new expressions. As such, it can be interpreted as a symbol-producing machine. The machine exists in a world of objects other than the symbols of the system. Designation and interpretation are central: the expressions designate objects, which means that the machine’s behavior can influence or be influenced by objects; the machine interprets expressions if it can carry out the processes that the expressions designate. These features are required for acting intelligently in the world.

A general-purpose computer is a machine with capabilities that form a symbol system. The previous assertion can be argued for with the development of CS [25]. The development of mathematical logic made it possible to manipulate symbols formally. The Turing machine and other computational models defined what can be computed and what cannot. Since Turing machine is capable of performing computations based on a description, it satisfies half of the principle of interpretation in a symbol system. The stored program concept of the von Neumann architecture gives the other half, since the program is part of the systems data and not given from outside. The invention of list processing gave data structures, that contained symbols and whose structure could be dynamically altered. This was an implementation of designation in accordance to the Physical Symbol System. The conclusion was that the computer has all the required features for intelligent action.

The properties of general-purpose computer can be used both as an implementation and model of intelligence [25]. The hypothesis stated that a physical symbol system is necessary and sufficient for intelligent action. The statement cannot be theoretically proven, it needs to be confirmed by experimental evidence [25]. Sufficiency can be shown by implementing a computer system that act’s intelligently. This is the research topic of Artificial intelligence that was first concerned with particular problems where solutions required intelligence. From that followed research that aimed to generalize the
mechanisms used in intelligent problem solving. Necessity can be shown by trying to explain intelligent action of which humans are the prime example. Research in Cognitive Psychology uses experiments and observations of behavior to understand intelligence. The computer is used as a model of the reason why certain situation or problem leads to a certain behavior. The benefit to traditional modelling is that these models can be executed, which allow for easier prediction of different kinds of scenarios.

V. WHAT IS UNIQUE IN COMPUTATIONAL THINKING

Computational thinking combines the theoretical, engineering and scientific perspectives. The perspectives look at the world differently; what problems can be solved, what tasks can be accomplished, or what features of the world can be understood computationally [33]. We have described above each perspective using seminal examples and the basic method of problem-solving belonging to the perspective. To reveal how they differ we look at each description using Shute et al. [14] components of Computational thinking: decomposition, abstraction, algorithm design, debugging, iteration and generalization. However, the goal of this paper is to find also, what is unique to Computational thinking, so we are looking at the same time for what is common to all the perspectives.

Alan Turing proved that the decision problem isn’t solvable by mechanizing the execution of an algorithm. Use of the Turing machine as a device for proof contains the six components of Computational thinking. First, the process of a human calculating on a paper was decomposed to consequent parts. Abstraction was utilized when the paper was substituted with an infinite squared tape, calculation was simplified to manipulating one symbol at a time and human thinking was replaced with rules. Algorithms played central role both as a tool and as the target of investigation. Both the creation of the Turing machine and the preparation of the proof had many iterations were every step was important in the deductive chain. Generalization was utilized when Turing made a machine that could emulate any other Turing machine. Debugging can certainly be imagined in this kind of process and corrections to the original paper was published [34] later.

The ENIAC project took what was known of mechanical and human calculation and turned it into an electronic form [30]. The electronic form made it possible to automate (a more full-ended form of mechanization) the computing process. The six components of Computational thinking can also be found in the engineering of ENIAC. The calculation was decomposed to follow how a human arranges his computation. The computer was an abstraction combining mechanical and human method of computing. Algorithms were central since the computation had to be constructed from simple operations. The engineering process was iterative with design, testing and debugging. Even though ENIAC was made specifically for calculating trajectories its design was general enough that it could be used for other computations which was lucky since the need for trajectory tables was diminished because the World War II ended.

Newell’s and Simon’s [25] use of artificial intelligence and models of cognition to illustrate CS as an empirical science is particularly appropriate since the criteria of intelligence is given from the outside. For comparison, in Software Engineering the method and the object of the investigation is not that easily separated [10]. In the scientific perspective, the six components of Computational thinking can be found in the Physical Symbol System hypothesis example. Decomposition is natural for investigating symbol systems since they are composed of symbols, expressions and processes that operate on them. The use of a machine as a model for intelligence is a mechanical abstraction of something neurological [35]. The processes in symbol systems are algorithmic since they are computational. Debugging is a part of artificial intelligence because problems in the results can also be due to programming errors. Iteration is natural for the scientific process were progress is in relation to what has been done before. Artificial intelligence started with creating solutions to particular problems, which later became generalized as mechanisms for creating intelligence. The examples give three different instantiations of the six components of CT summarized in Table II.

| TABLE II. MODEL OF CT COMPONENTS APPLIED TO EXAMPLES OF CS PERSPECTIVES |
|------------------|------------------|------------------|
| CT               | Turing machine (Theory) | ENIAC (Technical) | The physical symbol system (Science) |
| Decomposition    | computational steps | how human perform calculation | system (symbols, expressions, processes) |
| Abstraction      | tape/paper, symbol/number, rules/thought | combining human and mechanical method | mechanical/ neurological |
| Algorithm design | tool, target | operations | computations in machine and mind |
| Iterations       | proof process | desing, testing, debugging | scientific process |
| Debugging        | correcting algorithmic errors | design errors | AI* and cognitive model operation errors |
| Generalization   | universal machine | other uses than calculating trajectories | intelligence regards of implementati on |

* Artificial intelligence

However, making computation mechanical seems to be a common to all the processes described. The goal of the six components of the model is to make the problem solving mechanical. The application of the model to theoretical, engineering and scientific examples in this paper highlight the nature of the different perspectives. We have defined what is unique to problem solving in CT and illustrated how the processes of the perspectives differ related to CT.
This supports the teachers in applying Shute et al. [14] definition and model when planning on how to incorporate CT in teaching different topics.

VI. CONCLUSION

High hopes are placed on learning Computational thinking. The claim is that the thinking habits from CS are beneficial for any kind of problem solving and that they form a central part of skills needed in modern sciences. However, there is a diversity of definitions, interventions, assessments and models of CT for education. There is also a lack of generally agreed-upon definition of CT and its main components. This makes it difficult for teachers to incorporate CT in their teaching. In this paper we augmented an existing definition and model of CT. Based on theoretical, engineering and scientific perspective on CS we defined CT to be about mechanization of computation. This gives the teacher a goal of using CT and a context for applying the model. We applied the model to examples of the different perspectives to highlight how their problem solving differ. The teacher can use this as a supporting framework when applying CT to new topic.

REFERENCES