Refining an Understanding of Computational Thinking

Cynthia C. Selby
University of Southampton
Highfield
Southampton UK
44 (0) 2380 593475
C.Selby@soton.ac.uk

John Woollard
University of Southampton
Highfield
Southampton UK
44 (0) 2380 592998
J.Woollard@soton.ac.uk

Abstract

This paper identifies aspects of computational thinking and proposes a description that meets the needs of teachers and academics reflecting upon the developing UK curriculum. Since Jeanette Wing's use of the term computational thinking in 2006, various discussions have arisen seeking a robust definition of the phrase with little consensus. In order to facilitate consistent curriculum design and appropriate assessment, it is argued that a definition and description of the elements of computational thinking need to be identified. Criteria are developed for the objectives of a computational thinking definition, in accordance with the needs identified in the literature. The most frequently occurring terms, descriptions, and meanings used to characterise computational thinking are also identified in the literature. Using the criteria as a guide and the collected terms as the vocabulary, a definition of computational thinking is developed which incorporates the concepts of automation, abstraction, decomposition, algorithmic design, evaluation, and generalisation.

Keywords: Computational thinking, definition, abstraction, decomposition, algorithmic thinking, algorithmic design, generalization, evaluation

Introduction

The term 'computational thinking,' when used by Jeanette Wing (2006) in her call to make thinking like a computer scientist a fundamental skill for everyone, excited educators and academics. This presented an opportunity to promote computer science to a wider audience, but it also introduced a challenge. Wing did not precisely define the term and state exactly what the nature of 'computational thinking' for everyone. Since then, there have been attempts by authoritative individuals and groups (Barr & Stephenson, 2011; Denning,

2007; Grover & Pea, 2013; Guzdial, 2008; National Research Council [NRC], 2010) to derive a definition of computational thinking.

The aim of this review is to shed new light on the discussions that attempt to develop a definition of computational thinking with the objectives including: to define more narrowly, not more broadly; to bring an order to the criteria not necessarily to accommodate all viewpoints; to refine the definition to facilitate assessment; to retain the validity of work that has been done previously, such as the development of curricula; to separate a definition from those activities that might promote acquisition of computational thinking skills; and to separate a definition from those artefacts and activities that evidence the use of those skills.

Method

In an attempt to remain true to Wing's original vision, this literature search began with the seminal ACM article (Wing, 2006). During the following years, there was much activity around the topic (Denning, 2007; Guzdial, 2008; NRC, 2010; NRC, 2011; Wing, 2007; Wing, 2008). As the term became more accepted and the 'for everyone' manifesto generated interest, the focus shifted to curricula and classroom experiences (Barr & Stephenson, 2011; Bell, Andreae & Lambert, 2010; Brinda, Puhlmann & Schulte, 2009; Computing at School Working Group [CAS], 2012; Iyer, Baru, Chitta, Khan & Vishwanathan, 2010). All this time, other fields were also exploring their connections to computational thinking (Eisenberg, 2010; Lieu & Wang, 2010; Serafini, 2011; Zhang & Luo, 2012). In order to reflect this range of interest in deriving a definition for computational thinking, a broad selection of literature databases was explored. Where available, citation indices were searched to reveal more recent refinement of the original definition. The searched databases represent the crossover between mathematics, psychology, sciences, computer science, engineering, education, and computer science education research. They include the Web of Science and Citations, Engineering Village, Google Scholar and

Citations, IEEE Explore, ACM Digital Library, Lecture Notes in Computer Science, Compendex, PsycINFO, ERIC, and the British Education Index.

In an attempt to contribute to the development of a definition, the publications were analysed to discern the development, over time, of the phrase computational thinking. Descriptions and suggested definitions of computational thinking were identified in each publication. The terminology, common across these descriptions and definitions, was collated. Where equivalences allowed, similar terms were grouped together. Where descriptions were encountered without the direct usage of one of the proposed terms, they were attributed to the term. The most frequently occurring individual terms and groups of terms are presented in the following sections. From this basic collection of terms, a definition of computational thinking is formulated and proposed.

Justification for the inclusion or exclusion of terms is presented on a term-by-term basis. Justification is based on consistency of usage and consistency of interpretation across the literature. The resulting definition reflects much of the consensus found in the literature while removing less well-defined terms.

Evidence from literature

Some authors/papers/commentaries assert that a precise definition of computational thinking is not required (Guzdial, 2011; Hu, 2011). However, the discussion presented in this paper is driven by a perceived need to support professionals working in the field of computer science education and the developing computing curricula. This need for definition is supported in the literature (Barr & Stephenson, 2011; Cooper, Pérez & Rainey, 2010; Guzdial, 2011; NRC, 2011; NRC, 2010; Werner, Denner, Campe & Kawamoto, 2012).

Guzdial (2011) has suggested that a very broad definition is acceptable. Such acceptance could shift the focus away from what computational thinking is to how computational thinking should be taught and how evidence of its acquisition might be

observed in learners. Hu (2011) supports this by proposing that teachers are confident that the teaching of computer science does promote computational thinking. Even though they may not know exactly how this mechanism works, teachers recognise that the more learners practice computation, in terms of computer science, the better at computational thinking they become. Another broad definition of computational thinking is suggested by Grover and Pea (2013), in exploring the K-12 environment. This same argument is expressed by those who design or influence the design of computer science curricula. Several curricula (Bell et al., 2010; Brinda et al., 2009; Computer Science Teachers Association Task Force [CSTA], 2011; CAS, 2012) and cross-curricular activities (Curzon, Peckham, Taylor, Settle & Roberts, 2009; Eisenberg, 2010), while acknowledging the vagueness of a computational thinking definition, continue to include a focus on concepts and techniques from computer science. In presenting these concepts and techniques, the curricula include terminology often found in descriptions of computational thinking. Some of this terminology will be explored in more detail below.

Cuny (NRC, 2010) suggests that if computational thinking is included in a curriculum, it requires assessment. Without agreement on a common definition of computational thinking, it will be difficult, if not impossible, to develop appropriate assessment tools that actually measure the ability to think computationally (NRC, 2010). Werner et al. (2012) also view the lack of a definition and appropriate assessment tools as adversely affecting the delivery of computational thinking to secondary pupils. Current definitions of computational thinking are seen as confusing to those not trained in computer science (Cooper et al., 2010; Kranov, Bryant, Orr, Wallace & Zhang, 2010). A rigorous and agreed definition might ensure that computational thinking in these new curricula for the K-12 years will be more than, as Malyn-Smith argues, '… just a bunch of examples that are placed into the curriculum at the discretion of individual teachers' (NRC, 2011, p.33).

The balance of argument is still in favour of searching for a robust definition of computational thinking. Although it may be possible, without a robust definition, to identify examples of the practice of computational thinking, the ability to measure computational thinking may be hampered by that same lack.

Consensus terms

Three terms appear consistently throughout the literature reviewed here. There appears to be a consensus that a definition of computational thinking should include the idea of a thought process, the concept of abstraction, and the concept of decomposition.

A thought process

When introducing the term, computational thinking, Wing (2006) described it as a way that humans think about solving problems. It incorporates the set of mental tools used in computer science. These tools are used to transform a difficult problem into one that can be solved more easily. The Royal Society identifies computational thinking in an even broader sense as '… the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes' (The Royal Society, 2012, p. 29). In adding his voice to Wing's, calling for the explicit teaching of computational thinking, Guzdial (2008) refers to computational thinking as a way of thinking about computing. Workshop participants (NRC, 2010) agreed that it incorporates a range of mental tools and concepts from computer science. This idea is extended to represent problems as information processes and solutions as algorithms (Denning, 2011). Aho (2012) picks up the idea of problem transformation when he describes computational thinking as the thought processes in formulating problems and solutions that can be expressed as algorithms. These thought processes do have focus; frequently that focus is described as problem solving. Finally, Wing

expresses these refinements by defining computational thinking as '… the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent' (Cuny, Snyder & Wing, 2010, cited in Wing, 2011, p.20). Because of this consensus, a definition of computational thinking should include the concept of a thought process.

Abstraction

Although the idea of abstraction, hiding complexity, as being part of computational thinking is introduced by Wing in her original article (Wing, 2006), the definition develops over subsequent years. She amends the definition to include simultaneous consideration for multiple layers of abstraction and consideration for defining the interfaces between the layers (Wing, 2007). Even Denning (Ubiquity, 2007) acknowledges that abstraction plays an important part in computing, including programming. However, he points out that the act of abstracting is not unique to computer science. The next year, Wing (2008) defines abstraction as the cornerstone of computational thinking. Several participants concur that computational thinking has a focus around the process of abstraction, creating them and defining the relationships between them (NRC, 2010). More recently, Barr and Stephenson (2011) include the ability to abstract in a definition of computational thinking applicable to the K-12 age group. The concept of abstraction is explored by L'Heureux, Boisvert, Cohen and Sanghera (2012) where it is one of six aspects of their information technology approach to computational thinking. Lu and Fletcher (2009) view programming as analogous to literary analysis in English and propose the use of a computational thinking language, which includes the concept of abstraction. Again, in the middle and high school classroom, abstraction is identified as one of the key concepts of computational thinking (Lee et al., 2011; Yevseyeva & Towhidnejad, 2012). Higher education (Zhang & Luo, 2012) is also represented in this group advocating the inclusion of abstraction in a definition. In addition

to functional abstraction, data abstractions are also included (Voskoglou & Buckley, 2012). Because of this consensus, a definition of computational thinking should include the concept of abstraction.

Decomposition

Decomposition is required when dealing with large problems, complex systems, or complex tasks. Breaking problems down by functionality is identified by Wing (2006, 2007) as part of computational thinking. The participants in the first NRC workshop also identify the need for problem decomposition (NRC, 2010). In the next workshop, focusing on pedagogy, participants extend this idea. Tinker views the core of computational thinking as breaking down big problems (NRC, 2011). Edelson points out that the creation of solutions requires breaking problems down into chunks of particular functionality and sequencing the chunks (NRC, 2011). Most recently, in refining his own definition of computational thinking, Guzdial (2012) includes the use of tools including abstraction and decomposition. In the secondary classroom, Yevseyeva and Towhidnejad (2012) also use the term decomposition in their working definition of computational thinking. In light of this consensus, a definition of computational thinking should include the concept of decomposition.

Three terms are proposed for inclusion in the definition of computational thinking. Inclusion of a thought process, abstraction, and decomposition is supported by a consensus found in the reviewed literature. These terms are used consistently across the literature. Their use does not reflect any discrepancy in perceived meaning of the terms.

Other possible terms

Although less consistently than the terms above, several different terms and ideas do recur across the literature reviewed here. Even if a term or idea recurs, its interpretation is

not always consistent across articles. Several other proposed terms are broad and high-level. A lack of specific interpretation may make inclusion of these terms in a definition difficult. The terms identified fall into these four areas: thinking, problem solving, computer science, and imitation terms

Thinking terms

Although the idea that computational thinking represents a cognitive process attracts consensus, there are suggestions that specific types of thinking should be incorporated. These specific types of thinking are logical thinking, algorithmic thinking, engineering thinking, and mathematical thinking.

The concept of logical thinking, although not specifically defined, occurs several times in the literature. Albeit not perceived exactly as equivalent, terms to describe similar types of thinking are grouped into this category. These include mathematical thinking, engineering thinking, and heuristic thinking. In her original article, Wing (2006) indicates that computational thinking incorporates heuristic reasoning to devise a solution. In addition to abstraction and decomposition, Guzdial (2012) also includes heuristic reasoning as an appropriate tool to use when engaging in computational thinking. Logical reasoning is included by Iyer et al. (2010) in their model computer science curriculum in order to promote high-level thinking skills that are not necessarily subject specific. L'Heureux et al. (2012), in detailing an aspect of their information technology approach to computational thinking, define logical thinking as the ability to develop and test hypotheses.

Computational thinking also intersects with engineering because computer systems interact with the real world. However, computational thinkers can design and create virtual worlds, not limited by physical reality (Wing, 2007). Although Wing (2007) states that computer science relies on mathematics as a foundation, Sussman (NRC, 2010) affirms that mathematical thinking revolves around abstract structures while computational thinking

revolves around abstract methodology. The connections between computational thinking and mathematics, via the concepts in discrete mathematics, are proposed by Lieu and Wang (2010). Zhang and Luo (2012) propose that computational thinking is an integration of both mathematical and engineering thinking. In primary education, Eisenberg (2010) suggests exposing pupils to computational thinking with tactile objects including paper and beads. Computational thinking could be viewed as bringing science and engineering together. It could be viewed as a meta-science concerned with studying methods of thinking that are applicable to many different disciplines (NRC, 2010). While the ability to think logically, mathematically, heuristically, and from an engineering perspective are certainly capabilities that a computational thinker may exhibit, references to these terms in this literature are not well expanded.

Although the term logical thinking, as described above, may not be suitable to include in a definition of computational thinking, the potentially analogous term, algorithmic thinking, requires further investigation. In her original article, Wing (2006) does not use the term algorithmic thinking, preferring the word heuristic instead. However, by 2011, she extends her definition of computational thinking to include algorithmic and parallel thinking (Wing, 2011). Moursund (NRC, 2010) suggests that computational thinking is related to the idea of procedural thinking, as proposed by Seymour Papert in *Mindstorms*. He defines a procedure as a step-by-step set of instructions that can be carried out by a device. The same theme is continued by Sussman (NRC, 2010), who defines computational thinking as a way of devising explicit instructions for accomplishing tasks. The idea of algorithm is further extended to include the notion of basic flow control (Lu & Fletcher, 2009). Inclusion of algorithmic thinking in a curriculum for high schools appears prior to Wing's contribution. In the Israeli computer science curriculum, Gal-Ezer, Beeri, Harel, and Yehudai (1995) placed an emphasis on inclusion of the study of algorithmic processes. In primary schools,

the concept of algorithm is interpreted by Serafini (2011) to be key to computational thinking. Another example of the use of algorithmic thinking in the classroom is provided by Davies (2008) who advocates splitting thinking tasks from programming tasks. Contexts, other than computer science, are also identified as using the computational thinking concept of algorithmic thinking (Yevseyeva & Towhidnejad, 2012). The term algorithm is interpreted as a step-by-step procedure for accomplishing tasks, not just in computer science, but in other disciplines. It is evidenced through the creation of algorithms – algorithmic design. There appears to be a consensus that computational thinking incorporates aspects of the creation and use of algorithms. In order to represent these contributions to a definition of computational thinking, the single term algorithm design is proposed.

Not all of the types of thinking suggested for inclusion in the definition of computational thinking bring further refinement to the term. Tying a definition of computational thinking to terms such as logically or heuristically, with their open-ended interpretation, or to specific disciplines such as mathematics or engineering do not help advance the development of K-12 curricula and do not aid the development of computational thinking assessment instruments. Inclusion of these terms broadens the definition of computational thinking rather than narrows the focus. Terms expressing the idea of logical thinking or equivalence further dilute a definition of computational thinking. On the other hand, the idea of algorithm, incorporating the design process, is represented consistently in literature and its interpretation does not vary. Because of its wide acceptance and appropriate definition, the idea of algorithm is applicable for inclusion in a definition of computational thinking. Contributions from the literature that incorporate the idea of algorithm are represented by the term algorithmic design.

Problem solving terms

The idea that computational thinking has some relationship to problem solving appears frequently in the represented literature. The specific terms problem solving, analysis, and generalisation are most frequently employed in discussions of general problem-solving skills. This section explores the interpretation of these terms and the viability of incorporating them into the definition of computational thinking.

Problem solving, in one form or another, appears frequently in the literature presented here. There is agreement for describing computational thinking as a problem-solving activity. However, the literature does not illuminate problem solving in detail. Wing (2006, 2008), of course, incorporates solving problems using computer science concepts in her definition of computational thinking. The broadness of the problem-solving skills employed in computational thinking, in opposition to specific technical skills, is pointed out by Snyder (NRC, 2010). Kranov et al. (2010), and Voskoglou and Buckley (2012) identify a close relationship between computational thinking and critical thinking. A requirement for a computing device is introduced by Barr and Stephenson (2011), who state that the essence of computational thinking is solving problems in a way that can be implemented with a computer. Henderson (NRC, 2011) concisely describes computational thinking as a type of generalised problem solving with constraints. Problem solving is emphasised by Linn (NRC, 2010) who includes in the qualities of a successful computational thinker, the ability to engage in sustained investigative processes to generate problem solutions. Although there appears to be a consensus that computational thinking is perceived as a type of problem solving, the term is not sufficiently specific to define it.

The term analysis is included by some commentators in the definition of computational thinking. Interestingly, the term appears in relation to problems, solutions, and data, as in analyse a problem, analyse a solution, and analyse the data. Analyse, in the

context of problems, fits the category of problem solving, as defined above. Collection and analysis of data is proposed in a definition of computational thinking by Yevseyeva and Towhidnejad (2012). Analyse, in the context of solutions, could be interpreted as the comparable term evaluate. In her initial article, Wing (2006) expresses the need for a computational thinker to make trade-offs, by evaluating the use of time and space, power and storage. This evaluation of algorithmic processes, including their power and limitations, is foreshadowed by Gal-Ezer et al. (1995). Evaluation, in the guise of evaluating processes, is identified again by Lu and Fletcher (2009). Although the term analysis is cited as an example of computational thinking, the descriptions accompanying it more closely fit the term evaluation (Lee et al., 2011). Application of the term to user interfaces is evidenced in the second objective of the New Zealand proposed curriculum, as part of designing programs (Bell et al., 2010). In their IT approach, L'Heureux et al. (2012) include the ability to evaluate processes, in terms of efficiency and resource utilisation, and the ability to recognise and evaluate outcomes. Although the term analyse attracts some agreement for inclusion in a definition of computational thinking, descriptions of the term found in this literature imply an evaluative process. Analyse, in the context of problems and data, incorporates the previously defined terms of abstraction and decomposition. Descriptions of the term analyse, in the context of solutions, are attributed to the term evaluation.

A specific term that appears sparingly in the literature definitions is generalisation. Generalisation is the step of recognising how small pieces may be reused and reapplied to similar or unique situations. It is the ability to move from specific to broader applicability, for example, understanding how to draw a square by defining internal angles, then applying the same algorithm to produce an approximation of a circle. The ability to recognise parts of solutions that have been used in previous situations or that might be used in future situations is included by Kolodner in a definition of computational thinking (NRC, 2011). These parts,

or functional pieces, can be used to solve the current problem or combined in different ways to solve new problems (NRC, 2011). This concept of generalising processes is also described by Voskoglou and Buckley (2012). The term generalisation, itself, is described in a proposed curriculum as recognising common patterns and by sharing common features (CAS, 2012). The idea moves forward from decomposition, described above. This same idea is expressed as transfer of computational pattern use from a game scenario to a science scenario by Basawapatna, Koh, Repenning, Webb, and Marshall (2011). Although the exact term, generalisation, is used sparingly in the literature, the idea of recognising and reusing common parts of a solution or process is appropriate for inclusion in a definition of computational thinking.

Possible terms examined in this section include problem solving, analysis, and generalisation. Problem solving is a broad term that, although used consistently throughout the literature, is not well defined. Analysis, used in the context of a problem or data, is also a broad term, often incorporating the ideas of abstraction and decomposition. Analysis, used in the context of a solution, is analogous to evaluation and is used consistently in the literature. Although the term generalisation is used infrequently in the literature, there are descriptions of analogous processes that are attributable to the term. Therefore, from this set of possible terms, the ones used most consistently, with the least disparity of interpretation, and which refine the definition are the terms evaluation and generalisation.

Computer science terms

It is clear that computational thinking has a deep relationship with computer science. Some suggest specific computer science terminology be included in a definition of computational thinking. The specific terms include systems design, automation, and more general computer science concepts such as recursion and recovery through redundancy. This

section explores the viability of incorporating these terms into the definition of computational thinking.

Systems design, although not mentioned frequently, is still used to describe computational thinking. Designing systems based on concepts used in computer science is mentioned by Wing (2006). Again, this inclusion is foreshadowed by Gal-Ezer et al. (1995) who incorporate the study of the design and implementation of computing systems in their curriculum. One of Denning's Great Principles of Computing includes a category based on the design and building of software systems (Denning, 2007). He goes further in describing systems as one of the four core practices, in which computing professionals engage, along with programming, modelling, and innovating (Ubiquity, 2007). The focus in each of these cases is systems design as a product-oriented process. Systems design evidences the ability to think computationally, but does not necessarily define it.

A particular term, popularised by Wing in defining computational thinking, is automation. She connects the term to that of abstraction when discussing the mechanisation of abstraction layers and the relationships between them (Wing, 2007). Denning also acknowledges that this is what happens when programming (Ubiquity, 2007). Later, a stronger connection is made by Wing when defining computing as the 'automation of our abstractions' (2008, p. 3718). While it is acknowledged that automation has an important role in evidencing computational thinking, its suitability for inclusion in a definition must be viewed critically. Specifying that the result of computational thinking must be implementable by a computing device uniquely separates it from those terms applicable across other domains, such as logical thinking and mathematical thinking. However, there is some variation in the perception of the relationship between automation and computing device. One perspective on automation proposes the need for a computer, a digital computing device. In this view, the automation is a computer program, visualisation, file,

model, or representation created or interacted with by a user. The examples of automation provided by Barr and Stephenson (2011) include the use of spreadsheet applications, modelling software, and programming environments. The idea of models and simulations enabling automation is also suggested by the NRC (2010). This supports the idea that automations allow repetitive tasks to be undertaken with minimal human input (NRC, 2011; Lee et al., 2011). Both of these facilities, spreadsheets and multiple runs of simulations, are identified as automation by the CSTA (2011). Robotics is interpreted as automation by Grover and Pea (2013). Lee et al. (2011) describe interactions with automations rather than the creation of automations. The process or processes required in the creation of these automations may be possible terms for defining computational thinking. This is suggested by Voskoglou and Buckley (2012) when referring to the automating of solutions. Another perspective on automation as part of computational thinking asserts that there is no requirement for a digital computing device. In this view, the automation may be a process, idea, or even algorithm, but not one that must involve the use of a digital computing device. A program artefact, which is often viewed as automation, is only evidence that computational thinking has taken place. This is supported by Yevseyeva and Towhidnejad (2012) who conclude that computational thinking does not require the use of a computer. Wing (2008) goes to some length to include people when discussing mechanisation of abstractions. The idea of including a human computing device removes the need for an automation to be implementable by a digital device. Perković, Settle, Hwang, and Jones propose a definition of automation that acknowledges this subtlety, 'Automation is the mapping of computation to physical systems that perform them' (2010, p. 124). This leaves open the possibility of a human being the physical system that performs an automation. Therefore, while some authors (NRC, 2010; Barr & Stephenson, 2011; Lee et al., 2011; Grover & Pea, 2013) choose examples of automation that are closely associated with digital computing devices, there is no

limit on the physical form that an automation might take (Wing, 2008; Perković et al., 2010). A balance must be considered between the uniqueness that inclusion of the term automation brings to the definition of computational thinking and the tendency to interpret automation as a program, model, or visualisation requiring a digital computing device. Although including the term may necessitate distinctly addressing that a digital computing device is not needed, the uniqueness brought by the term to computational thinking mandates its inclusion in a definition.

Throughout the literature, terms closely related to the general content of computer science studies appear in descriptions of computational thinking. Wing (2007) herself introduces computer science concepts such as thinking recursively, interpreting code as data and data as code, type checking, prevention, detection, recovery through redundancy, damage containment, error correction, prefetching, and caching. Additional concepts such as parallel processing, testing, debugging, search strategies, algorithmic complexity, and pattern matching are recognised in the NRC report (2010). Barr and Stephenson (2011) include the abilities to think iteratively and recursively. Closer analysis reveals that not all of these concepts are unique to the field of computer science. For example, mathematicians think iteratively and engineers plan for recovery through redundancy. While each of these concepts may be mastered by computational thinkers, none of them uniquely defines or helps narrow a definition of computational thinking.

Possible terms examined in this section include systems design, automation, and more general computer science concepts such as recursion and recovery through redundancy. Systems design, resulting in a product, is evidence of the use of computational thinking skills, not a definition of it. Automation, as an implementation of abstractions by computing devices, uniquely distinguishes computational thinking from other forms of thinking. It is, however, important to acknowledge that computing devices are not limited to digital devices

and do include humans as computing devices. Including terms that are interpretable as computer science content, such as recovery through redundancy and parallel processing, do not bring focus to the definition of computational thinking. Therefore, from the selection of terms discussed in this section, only that of automation is suitable for inclusion in a definition of computational thinking.

Imitation terms

Three additional terms, also used in discussions of computational thinking, are modelling, simulation, and visualisation. These terms appear frequently in the represented literature. This section explores the viability of including these terms in a definition of computational thinking.

Wing (2006) began by defining computational thinking as modelling the appropriate parts of a problem to facilitate a solution. Later, Blake (NRC, 2010) insists that the definition of computational thinking should include modelling and visualisations. Brinda, Puhlmann, and Schulte (2009) have identified, as one achievable curriculum standard, the processes involved in modelling data. In the field of discrete mathematics, computer modelling is the mathematical and computer-based process of solving real world problems (Liu & Wang, 2010). On the other hand, Fox and Kolodner (NRC, 2010) point out that it is the manipulation of abstractions (models, simulations, and visualisations) that contribute to the development of computational thinking skills. Cooper, Pérez, and Rainey (2010) define this way of learning as 'computational learning.' In this context, a computer is a prerequisite for developing skills in STEM subjects. Observing the results of changing variable values, forming hypotheses, finding anomalies in data, and identifying invariants can all be achieved by interacting with models, simulations, and visualisations. The manipulation of these representations are agreed to enhance the development of computational thinking, but do not

necessarily define it. Although these tools are effective aids in developing computational thinking skills, they are not suitable for inclusion in a definition of computational thinking.

Possible terms examined in this section include modelling, simulation, and visualisation. These terms represent artefacts that evidence the use of computational thinking or tools used to enhance computational thinking. As such, they are excluded from a potential definition of computational thinking.

Proposed definition

The intent of this investigation is to shed new light on the discussions that attempt to develop a definition of computational thinking. The objectives for such a definition, as stated above, are: to define more narrowly, not more broadly; to bring an order to the criteria not necessarily to accommodate all viewpoints; to refine the definition to facilitate assessment; to retain the validity of work that has been done previously, such as the development of curricula; to separate a definition from those activities that might promote acquisition of computational thinking skills; and to separate a definition from those artefacts and activities that evidence the use of computational thinking skills. Justification for inclusion or exclusion is based on consistency of usage and consistency of meaning across the literature. Where equivalences allowed, similar terms were grouped together. Where descriptions were encountered without the direct usage of one of the proposed terms, they were attributed to the term. The resulting definition reflects much of the consensus found in the literature while removing the less well-defined terms.

Table 1 summarises the justification for each prospective term's inclusion in or exclusion from a proposed definition of computational thinking.

Term	**Status**	**Justification**
A thought process	Include	Consensus found in the literature

Abstraction	Include	Consensus found in the literature
Decomposition	Include	Consensus found in the literature
Logical thinking	Exclude	Broad term, not-well defined; incorporates the concepts of abstraction and decomposition
Algorithmic design	Include	Well-defined across multiple disciplines; adheres to idea of a thought process; does not require creation of algorithm limited to digital computing devices
Problem solving	Exclude	Broad term; evidences the use of skills; develops acquisition of skills
Analysis	Exclude	Broad term; incorporates concepts of abstraction, decomposition, algorithmic design, and evaluation
Evaluation	Include	Well-defined across multiple disciplines
Generalisation	Include	Well-defined concept; often encountered descriptively in the literature
Systems design	Exclude	Evidences the use of skills
Automation	Include	Distinct from other forms of thinking; must not be interpreted as requiring a digital device
Computer science content	Exclude	Evidences the use of skills
Modelling, simulation, and visualisation	Exclude	Evidences the use of skills in their creation; manipulation develops acquisition of skills

Table 1. Computational thinking definition terminology

As supported by the preceding arguments, computational thinking is an activity associated with problem solving, often resulting in an artefact or product. Computational thinking is a cognitive process resulting in an automation that is developed by the use of abstraction, decomposition, algorithmic design, evaluation, and generalisation.

Conclusion

There is a need for a robust, clear, and agreed definition of computational thinking. The definition, to gain both academic and professional credence, should facilitate the development of computer science curricula in line with Wing's original vision to encourage computational thinking for all. The definition also needs to support classroom practice and curriculum development where the teaching of computing is undergoing radical changes from K-12. The review of the literature and analysis of the findings bring further understanding of the complex nature of the construct 'computational thinking' that has bearing upon the teaching of learners of all ages. The definition is refined by exclusion and inclusion of specific terms by the application of selection criteria. The definition can now enable appropriate assessment tools to be developed which measure computational thinking skills. In conclusion, computational thinking is a brain-based activity that enables problems to be resolved, situations better understood, and values better expressed through systematic application of abstraction, decomposition, algorithmic design, generalisation, and evaluation in the production of an automation implementable by a digital or human computing device.

References

Aho, A. V. (2012). Computation and Computational Thinking. *Computer Journal*, 55(7)**,** 832-835. doi:10.1093/comjnl/bxs074.

Barr, V. & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What Is Involved and What Is the Role of the Computer Science Education Community? *ACM Inroads,* 2(1)**,** 48-54. doi:10.1145/1929887.1929905.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C. & Marshall, K. S. (2011). Recognizing Computational Thinking Patterns. *Proceedings of the 42nd ACM*

Technical Symposium on Computer Science Education, 245-250. New York: ACM. doi:10.1145/1953163.1953241.

Bell, T., Andreae, P. & Lambert, L. (2010). Computer Science in New Zealand High Schools. *Proceedings of the Twelfth Australasian Conference on Computing Education*, 103, 15-22. Australian Computer Society, Inc.

Brinda, T., Puhlmann, H. & Schulte, C. (2009). Bridging ICT and CS - Educational Standards for Computer Science in Lower Secondary Education. *Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, 288-292. New York: ACM. doi:10.1145/1562877.1562965.

Computer Science Teachers Association Task Force (2011) *K–12 Computer Science Standards.* New York, ACM.

Computing at School Working Group. (2012) *Computer Science: A Curriculum for Schools.* Retrieved from: http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf.

Cooper, S., Pérez, L. C. & Rainey, D. (2010). K--12 Computational Learning. *Commun. ACM*, 53(11), 27-29. doi:10.1145/1839676.1839686.

Curzon, P., Peckham, J., Taylor, H., Settle, A. & Roberts, E. (2009). Computational Thinking (CT): On Weaving It In. *SIGCSE Bull.*, 41(3), 201-202. doi:10.1145/1595496.1562941.

Davies, S. (2008). The Effects of Emphasizing Computational Thinking in an Introductory Programming Course. *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, IEEE. doi:10.1109/fie.2008.4720362.

Denning, P. J. (2007). Computing Is a Natural Science. *Commun. ACM*, 50(7), 13-18. doi:10.1145/1272516.1272529.

Denning, P. J. (2011). Ubiquity Symposium: What Have We Said About Computation?: Closing Statement. *Ubiquity*, 2011(April), 1-7. doi:10.1145/1967045.1967046.

Eisenberg, M. (2010). Bead Games, or, Getting Started in Computational Thinking without a Computer. *International Journal of Computers for Mathematical Learning*, 15(2), 161-166. doi:10.1007/s10758-010-9167-5.

Gal-Ezer, J., Beeri, C., Harel, D. & Yehudai, A. (1995). A High School Program in Computer Science. *Computer*, 28(10), 73-80. doi:10.1109/2.467599.

Grover, S. & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38-43. doi:10.3102/0013189x12463051.

Guzdial, M. (2008). Education: Paving the Way for Computational Thinking. *Commun. ACM*, 51(8), 25-27. doi:10.1145/1378704.1378713.

Guzdial, M. (2011, March 22). Re: *A Definition of Computational Thinking from Jeannette Wing*. [Web log message]. Retrieved from http://computinged.wordpress.com/2011/03/22/a-definition-of-computational-thinking-from-jeanette-wing/.

Guzdial, M. (2012, April 6). Re: *A Nice Definition of Computational Thinking, Including Risks and Cyber-Security*. [Web log message]. Retrieved from http://computinged.wordpress.com/2012/04/06/a-nice-definition-of-computational-thinking-including-risks-and-cyber-security/.

Hu, C. (2011). Computational Thinking: What It Might Mean and What We Might Do About It. *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 223-227. New York: ACM. doi:10.1145/1999747.1999811.

Iyer, S., Baru, M., Chitta, V., Khan, F. & Vishwanathan, U. (2010) *Model Computer Science Curriculum for Schools.* Retrieved from: http://www.cse.iitb.ac.in/~sri/papers/CSC-April2010.pdf.

Kranov, A. A., Bryant, R., Orr, G., Wallace, S. A. & Zhang, M. (2010). Developing a Community Definition and Teaching Modules for Computational Thinking: Accomplishments and Challenges. *2010 ACM Conference on Information Technology Education, SIGITE 2010***,** 143-148. New York: ACM. doi:10.1145/1867651.1867689.

L'Heureux, J., Boisvert, D., Cohen, R. & Sanghera, K. (2012). It Problem Solving: An Implementation of Computational Thinking in Information Technology. *Proceedings of the 13th Annual Conference on Information Technology Education***,** 183-188. New York: ACM. doi:10.1145/2380552.2380606.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads,* 2(1)**,** 32-37. doi:10.1145/1929887.1929902.

Liu, J. & Wang, L. (2010). Computational Thinking in Discrete Mathematics. *Second International Workshop on Education Technology and Computer Science (ETCS),* 1**,** 413-416. IEEE. doi:10.1109/etcs.2010.200.

Lu, J. J. & Fletcher, G. H. L. (2009). Thinking About Computational Thinking. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education***,** 260-264. New York: ACM. doi:10.1145/1508865.1508959.

National Research Council. (2010) *Report of a Workshop on the Scope and Nature of Computational Thinking.* Retrieved from: http://www.nap.edu/catalog.php?record_id=12840.

National Research Council. (2011) *Report of a Workshop of Pedagogical Aspects of Computational Thinking.* Retrieved from: http://www.nap.edu/catalog.php?record_id=13170.

Perković, L., Settle, A., Hwang, S. & Jones, J. (2010). A Framework for Computational Thinking across the Curriculum. *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education***,** 123-127. New York: ACM. doi:10.1145/1822090.1822126.

Serafini, G. (2011) Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects, In: I. Kalaš & R. Mittermeir (Eds) *Informatics in Schools. Contributing to 21st Century Education.* (vol. 7013), 143-154, Berlin: Springer-Verlag.

The Royal Society. (2012) *Shut Down or Restart? The Way Forward for Computing in UK Schools.* Retrieved from: http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf.

Ubiquity (2007). An Interview with Peter Denning on the Great Principles of Computing. *Ubiquity,* 2007(June)**,** 1. doi:10.1145/1276162.1276163.

Voskoglou, M. G. & Buckley, S. (2012). Problem Solving and Computational Thinking in a Learning Environment. *Egyptian Computer Science Journal,* 36(4)**,** 28-46. doi:arXiv:1212.0750v1.

Werner, L., Denner, J., Campe, S. & Kawamoto, D. C. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. *Proceedings of the 43rd ACM technical symposium on Computer Science Education***,** 215-220. New York: ACM. doi:10.1145/2157136.2157200.

Wing, J. (2006). Computational Thinking. *Commun. ACM,* 49(3)**,** 33-35. doi:10.1145/1118178.1118215.

Wing, J. (2007) *Computational Thinking.* Retrieved from: http://www.cs.cmu.edu/afs/cs/usr/wing/www/Computational_Thinking.pdf.

Wing, J. (2008). Computational Thinking and Thinking About Computing. *Philosophical Transactions of The Royal Society A,* 366**,** 3717-3725. doi:10.1098/rsta.2008.0118.

Wing, J. (2011) *Research Notebook: Computational Thinking - What and Why?* Retrieved from: http://link.cs.cmu.edu/files/11-399_The_Link_Newsletter-3.pdf.

Yevseyeva, K. & Towhidnejad, M. (2012). Work in Progress: Teaching Computational Thinking in Middle and High School. *Frontiers in Education Conference (FIE), 2012***,** 1-2. IEEE. doi:10.1109/fie.2012.6462487.

Zhang, Y. & Luo, C. (2012). Training for Computational Thinking Capability on Programming Language Teaching *The 7th International Conference on Computer Science & Education (ICCSE 2012)***,** 1804-1809. IEEE. doi:10.1109/ICCSE.2012.6295420