Computational Thinking: The Developing Definition

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

ABSTRACT
Since Jeanette Wing's use of the term computational thinking in 2006, various discussions have arisen seeking a robust definition of the phrase. With no consensus having been found in the intervening years, there are even suggestions that a definition is not important. Perhaps focus should be on how computational thinking is taught and how its acquisition might be observed. However, in order to facilitate consistent curriculum design and appropriate assessment, it is argued that a definition should still be sought.

In order to contribute to the discussions surrounding a definition of computational thinking, this review of literature spans the years since 2006. The most frequently occurring terms, descriptions, and meanings are identified. Consideration is given to the motivation for inclusion or exclusion of a term by each individual author. Where possible, if a description has been given, an associated term is supplied.

Criteria are developed for the objectives of a computational thinking definition, in accordance with the needs identified in the literature. Using the criteria as a guide and the collected terms as the vocabulary, a definition of computational thinking is proposed.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computers and Education, Curriculum

General Terms
Standardization, Theory

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

1. INTRODUCTION
The term "computational thinking," when used by Jeanette Wing [19] in her call to make thinking like a computer scientist a fundamental skill for everyone, excited educators. This presented an opportunity to promote computer science to a wider audience, but also introduced a problem. She did not precisely define the term. What exactly is this "computational thinking" for everyone?

Since then, there have been attempts by authoritative individuals and groups [1, 16, 9, 6] to derive a definition for computational thinking. The development of a definition, as found in the literature is recounted in following sections, but the selection process for the literature follows immediately.

A selection of literature applicable to the topic of computational thinking was refined using the following method. An Internet search engine query using the criteria "Jeannette Wing" AND "computational thinking" was initially executed. The first four pages of results were interpreted for applicability of title. All documents identified as having applicable titles were individually inspected. This resulted in six individual documents. The ACM Digital Library was searched using the term "Jeannette Wing" in the author field. An additional search was made using "Jeannette Wing" in any field. Both searches were refined to return articles published since 2005. All results were inspected first for title suitability. All suitably titled articles were then refined by abstract, if provided. If no abstract was provided, then the first page of the article was read. This led to the identification of thirteen articles and reports. Curriculum designs, specifically for K-12 learners, were sought next. This search highlighted proposed or current computer science curriculums in Israel, Germany, New Zealand, India, England, and the USA. This search led to the identification of seven appropriate documents. Four of the total potential items, including some video lectures, were repetitions of an author's previous comments. Therefore, of the initially identified twenty-six documents, twenty-two were selected for this investigation.

2. EVIDENCE FROM LITERATURE
Although some authors/papers/commentaries may assert that a precise definition of computational thinking is not required [10, 13], others value such a description for a number of reasons [1, 17, 16]. The reform of computer science education, in line with Wing's vision, will necessitate the creation of curriculums that address the broader issue of computational thinking as well as computer science concepts. A rigorous and agreed definition might ensure that computational thinking in these new curriculums for the K-12 years will be more than, as Joyce Malyn-Smith argued, "… just a bunch of examples that are placed into the curriculum at the discretion of individual teachers" [17, p.33]. Further, Jan Cuny suggests that once 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 [16].

Proposals have been made to suggest that the time has come to move on from defining computational thinking. Guzdial [10] has suggested that a very broad definition is acceptable. More

importantly, the focus should now be shifted away from what computational thinking is to how computational thinking should be taught and how evidence of its acquisition might be observed in learners. Professor of Computer Science, Chenglie Hu [13], goes further. He suggests 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 know that the more learners practice computation, in terms of computer science, the better at computational thinking they become. Assuming that this is true, perhaps the focus should be on the practice of computing rather than on defining computational thinking. This same argument is expressed by some of those who design or influence the design of computer science curriculums. Several computer science curriculums [5, 4, 2, 3] 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 curriculums include terminology often found in descriptions of computational thinking. In an attempt to resolve some of the inconsistencies, these descriptive terms will be discussed in more detail below.

The balance of argument is still in favor of searching for a robust definition of computational thinking. To that end, the publications, selected as indicated above, were read in chronological order to discern the development of the phrase, computational thinking, over time. Descriptions and suggested definitions of computational thinking were identified in each publication. The terminology, common across descriptions and definitions, was collected. Where interpretation allowed, similar terms were grouped together. The most frequently occurring individual terms and groups 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. The resulting definition reflects much of the consensus found in the literature while removing the less well-defined terms.

3. REFLECTION ON CONSENSUS

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. Support for inclusion of these terms in a definition of computational thinking is presented in this section.

When introducing the term, computational thinking, Wing [19] 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. In adding his voice to Wing's, calling for the explicit teaching of computational thinking, Guzdial [9] refers to computational thinking as a way of thinking about computing. Participants in the workshop on the scope and nature of computational thinking [16], although not tasked with defining computational thinking, nevertheless 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 [7]. Al Aho [7] 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 [22], p.20). Because of this consensus, a definition of computational thinking should include the concept of a thought process.

Although the idea of abstraction, hiding complexity, as being part of computational thinking is introduced by Wing in her original article [19], it expands over the next few years. She amends the definition to include simultaneous consideration for multiple layers of abstraction and consideration for defining the interfaces between the layers [20]. Even Peter Denning [18] 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 [21] defines abstraction as the cornerstone of computational thinking. Several participants in the workshop on the scope and nature of computational thinking (NRC) concur that computational thinking has a focus around the process of abstraction, creating them and defining the relationships between them [16]. More recently, in their report on workshops sponsored by the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) to incorporate computational thinking into the K-12 curriculum, Barr and Stephenson [1] also include the ability to abstract in a definition of computational thinking. The concept of abstraction is explored by L'Heureux et al. [15] where it is one of six aspects of their information technology approach to computational thinking. Because of this consensus, a definition of computational thinking should include the concept of abstraction.

Breaking problems down by functionality is identified by Wing [19, 20] as part of computational thinking. Decomposition is required when dealing with large problems, complex systems, or complex tasks. The participants in the first NRC workshop also identify the need for problem decomposition [16]. In the next workshop, focusing on pedagogy, participants extend this idea. Robert Tinker views the core of computational thinking as breaking down big problems [17]. Danny Edelson points out that the creation of solutions requires breaking problems down into chunks of particular functionality and sequencing the chunks [17]. Most recently, in refining his own definition of computational thinking, Guzdial [11] includes the use of tools including abstraction and decomposition. 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. Although consensus has been demonstrated for these terms, others receive less support and more varied interpretation. Some of these additional terms and their applicability for inclusion in a definition of computational thinking are discussed below.

4. REFLECTION ON CANDIDATE 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 ideas proposed as part of a definition for computational thinking are broad and high-level. A lack of specific interpretation may make inclusion of these terms in a definition difficult. These terms include logical thinking; problem solving; algorithmic thinking; analysis; systems design; computer science

thinking; generalization; automation; and modeling, simulation, and visualization. Support for inclusion or exclusion of these terms in a definition of computational thinking is presented in this section.

The concept of logical thinking, although not specifically defined, occurs several times in the literature spanning these years. 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 [19] indicates that computational thinking incorporates heuristic reasoning to devise a solution. In addition to abstraction and decomposition, as described previously, Guzdial [11] also includes heuristic reasoning as an appropriate tool to use when engaging in computational thinking. Computational thinking is equivalent to the logical reasoning used by people [12]. Logical reasoning is included by Iyer et al. [14] in their model computer science curriculum in order to promote high-level thinking skills that are not necessarily subject specific. L'Heureux et al. [15], 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 [20]. Although Wing [20] states that computer science relies on mathematics as a foundation, Gerald Sussman [16] affirms that mathematical thinking revolves around abstract structures while computational thinking revolves around abstract methodology. 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 [16]. 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. Tying a definition of computational thinking to other terms such as logically or heuristically, with their open-ended interpretation, or to specific disciplines such as mathematics or engineering may not help advance the development of K-12 curriculums and may not aid in the development of computational thinking assessment instruments. For these reasons, terms expressing the idea of logical thinking or equivalence may dilute a 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 [19, 21], 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 Larry Snyder [16]. A requirement for a computing device is introduced by Barr and Stephenson [1], who state that the essence of computational thinking is solving problems in a way that can be implemented with a computer. Peter Henderson [17] concisely describes computational thinking as a type of generalized problem solving with constraints. Problem solving is emphasized by Marcia Linn [16] who includes in the qualities of a successful computational thinker, the ability to engage in sustained investigative processes to generate problem solutions. As stated above, computational thinking is a focused process. The focus is often product oriented in response to an issue, context, or problem. Although there appears to be a consensus that computational thinking is a type of problem solving, the term may not be sufficiently specific to define it. Computational thinking may be a subset of problem solving, not defined by it. Due to the broadness of the term, problem solving may not be suitable for inclusion in a definition of computational thinking.

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 [19] 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 [22]. David Moursund [16] 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 Gerald Sussman [16], who defines computational thinking as a way of devising explicit instructions for accomplishing tasks. Inclusion of algorithmic thinking in a curriculum for high schools appears prior to Wing's contribution. In the Israeli computer science curriculum, Gal-Ezer et al. [8] placed an emphasis on inclusion of the study of algorithmic processes. There appears to be a consensus that computational thinking incorporates aspects of algorithmic thinking. The term algorithm is interpreted as a step-by-step procedure for accomplishing tasks, not just in computer science, but in other disciplines. Because of its wide acceptance and appropriate definition, algorithmic thinking may be applicable for inclusion in a definition of computational thinking.

The term analysis is included by some commentators in the definition of computational thinking. Interestingly, the term appears in relation to both problems and solutions, as in analyze a problem and analyze a solution. Analyze, in the context of problems, fits the category of problem solving, as defined above. However, analyze, in the context of solutions, could be interpreted as the comparable term evaluate. In her initial article, Wing [19] 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. [8]. Application of the term to user interfaces is evidenced in the second objective of the New Zealand proposed curriculum, as part of designing programs [2]. In their IT approach, L'Heureux et al. [15] include the ability to evaluate processes, in terms of efficiency and resource utilization, and the ability to recognize and evaluate outcomes. Although the term analyze attracts some agreement for inclusion in a definition of computational thinking, descriptions of the term found in this literature imply an evaluative process. Therefore, because of consensus in the description, the term evaluate may be suitable for inclusion in a definition of computational thinking.

Systems design, although less frequently mentioned, is still used to describe computational thinking. Designing systems based on concepts used in computer science is mentioned by Wing [19]. Again, this inclusion is foreshadowed by Gal-Ezer et al. [8] who incorporates the study of the design and implementation of computing systems in their curriculum. One of Peter Denning's Great Principles of Computing includes a category based on the design and building of software systems [6]. He goes further in describing systems as one of the four core practices, in which computing professionals engage, along with programming, modeling, and innovating [18]. The focus in each of these cases is systems design as a product oriented process. It is evidence of the ability to think computationally, not necessarily a definition of it.

Therefore, the term systems design may not be suitable for inclusion in a definition of computational thinking.

Throughout the literature, terms closely related to the content of computer science studies appear in descriptions of computational thinking. Wing [20] 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 recognized in the NRC report [16]. Barr and Stephenson [2] include the abilities to think iteratively and recursively. 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. Therefore, terms interpretable as computer science content may not be helpful in defining computational thinking.

A specific term that appears sparingly in the literature definitions is generalization. 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 recognize 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 [17]. These parts, or functional pieces, can be used to solve the current problem or combined in different ways to solve new problems [17]. The term generalization, itself, is described in a proposed curriculum as recognizing common patterns and by sharing common features [5]. The idea moves forward from decomposition, described above. Generalization is the step of recognizing how small pieces may be reused and reapplied to similar or unique problems. Although the exact term, generalization, is used sparingly in the literature, the idea of recognizing and reusing common parts of a solution is a candidate for inclusion in a definition of computational thinking.

Another term, popularized by Wing in defining computational thinking, is automation. She connects the term to that of abstraction when discussing the mechanization of abstraction layers and the relationships between them [20]. Even Denning acknowledges that this is what happens when programming [18]. Later, a stronger connection is made by Wing [21] when defining computing as the "automation of our abstractions" (p. 3718). This introduces the need for a computational device to interpret the abstractions, the need for a computer to execute a program. The process or processes required in the creation of these automations may be candidates for defining computational thinking. On the other hand, a program artifact, similar to system design as discussed above, is only evidence that computational thinking has taken place. Previously, a consensus was presented that emphasized the thought process aspect of computational thinking. Based on that consensus, automation, interpreted as a program artifact, may not be a useful addition to the definition of computational thinking.

Three additional terms, also used in discussions of computational thinking, are modeling, simulation, and visualization. Wing [19] began by defining computational thinking as modeling the appropriate parts of a problem to facilitate a solution. Later, Brian Blake [16] insists that the definition of computational thinking should include modeling and visualizations. Brinda, Puhlmann, and Schulte [3] have identified, as one achievable curriculum standard, the processes involved in modeling data. On the other hand, Edward Fox and Janet Kolodner [16] point out that it is the manipulation of abstractions (models, simulations, and visualizations) that contribute to the development of computational thinking skills. 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 visualizations. The manipulation of these representations are agreed to enhance the development of computational thinking skills, but do not necessarily define it. In parallel with automation, the terms model, simulation, and visualization may not be suitable for inclusion in a definition of computational thinking.

A diverse group of terms has been presented in this section. Each of these terms has been employed in the literature in attempting to define and describe computational thinking. Support for the inclusion or exclusion of the term in the definition of computational thinking has been investigated. The following section summarizes the arguments presented above and suggests a definition of computational thinking based on these arguments.

5. PROPOSED DEFINITION

The intent of this investigation is to shed new light on the discussions that attempt to develop a definition of computational thinking. Justification for a definition is presented in a previous section. Based on an assumption that a definition is required, the objectives for such a definition should be considered. In the case of this investigation, the objects include: 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 curriculums; to separate a definition from those activities that might promote acquisition of computational thinking skills; and to separate a definition from those artifacts and activities that evidence the use of those skills. Table 1 summarizes 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
Problem solving	Exclude	Broad term, evidences the use of skills, develops acquisition of skills
Algorithmic thinking	Include	Well-defined across multiple disciplines
Evaluation	Include	Well-defined across multiple disciplines
Systems design	Exclude	Evidences the use of skills
Computer science content	Exclude	Evidences the use of skills

Generalization	Include	Well-defined concept, although the term may not be familiar
Automation	Exclude	Evidences the use of skills
Modeling, simulation, and visualization	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, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects

- the ability to think in abstractions,
- the ability to think in terms of decomposition,
- the ability to think algorithmically,
- the ability to think in terms of evaluation, and
- the ability to think in generalizations.

This proposed definition attempts to incorporate only those terms for which there is a consensus in the literature or those terms that are well defined across disciplines. The intent is to focus on the thinking aspect of the original phrase.

In other words, computational thinking is a focused approach to problem solving, incorporating thought processes that utilize abstraction, decomposition, algorithms, evaluation, and generalizations. It is closely associated with, but not defined by, the physical or applied skills of modeling, simulation, and visualization.

6. CONCLUSION

There is a genuine need, as discussed previously, for a robust and agreed definition of computational thinking. Such a definition may facilitate the development of computer science curriculums in line with Wing's original vision to encourage computational thinking for all. Such a definition may also ensure that the K-12 curriculums will not become just a collection of interesting resources presented at teachers' discretions. Such a definition may ensure that appropriate assessment tools can be developed which measure computational thinking skills. The description, as proposed above, narrows the definition by excluding some proposed terms. It separates the practice of skills from the thinking. It separates the results or evidence of the application of skills from the activity of thinking. However, it does not invalidate the curriculum designs, especially as they often focus on the doing or evidence of doing computational thinking. It leaves open the possibilities to develop assessment tools to measure the ability to think computationally. Of course, the discussions of a definition for computational thinking are not yet concluded. It may well be that the definition changes as understanding of computational thinking develops over the coming years. This is especially true as younger learners are exposed to the concepts in fulfillment of Wing's original vision of computational thinking for all. This review of the literature simply attempts to inform these discussions.

7. REFERENCES

[1] 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**,** 48-54.

[2] Bell, T., Andreae, P. & Lambert, L. 2010. Computer Science in New Zealand high schools. *Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103.* Brisbane, Australia: Australian Computer Society, Inc.

[3] 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.* Paris, France: ACM.

[4] Computer Science Teachers Association Task Force. 2011. *K–12 Computer Science Standards,* New York, ACM.

[5] Computing at School Working Group. 2012. Computer Science: A curriculum for schools. Available: http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf [Accessed 26-12-2012].

[6] Denning, P. J. 2007. Computing is a natural science. *Commun. ACM,* 50**,** 13-18.

[7] Denning, P. J. 2011. Ubiquity symposium: What have we said about computation?: closing statement. *Ubiquity,* 2011**,** 1-7.

[8] Gal-Ezer, J., Beeri, C., Harel, D. & Yehudai, A. 1995. A high school program in computer science. *Computer,* 28**,** 73-80.

[9] Guzdial, M. 2008. Education: Paving the way for computational thinking. *Commun. ACM,* 51**,** 25-27.

[10] Guzdial, M. 2011. A Definition of Computational Thinking from Jeannette Wing. *Computing Education Blog* [Online]. Available from: http://computinged.wordpress.com/2011/03/22/a-definition-of-computational-thinking-from-jeanette-wing/ [Accessed 22-03-11].

[11] Guzdial, M. 2012. A nice definition of computational thinking, including risks and cyber-security. *Computing Education Blog* [Online]. Available from: http://computinged.wordpress.com/2012/04/06/a-nice-definition-of-computational-thinking-including-risks-and-cyber-security/ [Accessed 06-04-12].

[12] Henderson, P. B., Cortina, T. J. & Wing, J. M. 2007. Computational thinking. *Proceedings of the 38th SIGCSE technical symposium on Computer science education.* Covington, Kentucky, USA: ACM.

[13] 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.* Darmstadt, Germany: ACM.

[14] Iyer, S., Baru, M., Chita, V., Khan, F. & Vishwanathan, U. 2010. Model Computer Science Curriculum for Schools. Available: http://www.cse.iitb.ac.in/~sri/papers/CSC-April2010.pdf [Accessed 28-12-2012].

[15] 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.* Calgary, Alberta, Canada: ACM.

[16] National Research Council. 2010. Report of a Workshop on the Scope and Nature of Computational Thinking. Available: http://www.nap.edu/catalog.php?record_id=12840 [Accessed 10-05-2011].

[17] National Research Council. 2011. Report of a Workshop of Pedagogical Aspects of Computational Thinking. Available:

http://www.nap.edu/catalog.php?record_id=13170 [Accessed 10-10-2011].

[18] Ubiquity. 2007. An Interview with Peter Denning on the great principles of computing. *Ubiquity*, 2007**,** 1-1.

[19] Wing, J. 2006. Computational thinking. *Commun. ACM*, 49**,** 33-35.

[20] Wing, J. 2007. *Computational Thinking* [Online]. Available: http://www.cs.cmu.edu/afs/cs/usr/wing/www/Computational_Thinking.pdf [Accessed 14-12-12].

[21] Wing, J. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society A*, 366**,** 3717-3725.

[22] Wing, J. 2011. Research Notebook: Computational Thinking - What and Why? *The Link.* Pittsburgh, PA: Carneige Mellon.