ACADEMIA | Letters

Thoughts on Computational Thinking

Maria do Carmo Nicoletti, Centro Universitário Campo Limpo Paulista (UNIFACCAMP)
Elisa Suemasu, Centro Universitário Campo Limpo Paulista (UNIFACCAMP)

The term *computational thinking* can be found in many references with focus on computing and related issues. Despite the term being recurrently employed in numerous scientific research works, it is considered that this line of thought has its origins in the pioneering work carried out by W. Feurzeig, S. Papert and C. Solomon who, in 1967, designed an educational programming language called Logo [Logo-foundation 2021, Goldberg 1982]. The main goal of the Logo language project was to provide a language and a computational environment that could stimulate and promote the way children think and solve problems, by means of activities that direct, encourage and strengthen knowledge learning. The term computational thinking was initially used by Papert in his book [Papert 1980], without a formal definition. Papert argues about the benefits of teaching computer literacy in primary and secondary education and states that "The cultural assimilation of the computer presence will give rise to a computer literacy. This phrase is often taken as meaning knowing how to program, or knowing about the varied uses made of computers. But true computer literacy is not just knowing how to make use of computers and computational ideas. It is knowing when it is appropriate to do so."

The Logo project as a whole was based on the research work developed by J. Piaget [Piaget 1952], referred to as the theory of cognitive development (also known as developmental stage theory), which focuses on the nature and development of human intelligence [Piaget 1952, Wadsworth 1971, Munari 1994]. In a simplistic approach, the Piaget's proposal for human cognitive development is divided into four stages. Adopting the notation *stage name/age/goal,* where *age* corresponds to the period each stage takes place, the stages are: (1) sensorimotor/from birth to 18-24 months old/object permanence; (2) preoperational/2-7

Corresponding Author: Maria do Carmo Nicoletti, carmocrat@gmail.com
Citation: Nicoletti, M.D.C., Suemasu, E. (2021). Thoughts on Computational Thinking. *Academia Letters*, Article 1045. https://doi.org/10.20935/AL1045.

years old/symbolic thought; (3) concrete operational/7-11 years old/operational thought, and (4) formal operational/adolescence to adulthood/abstract concepts. Obviously the human cognitive development through the years should be taken into consideration, when the goal is the proposal of a teaching-learning methodological strategy that focuses on motivating and supporting the development of any sort of computational thinking, taking into account the several different semantics associated to the term.

Authors in [Yadav et al. 2014] inform that computational thinking has a long history in the context of Computer Science (CS) and that its origins can be dated back to the 50s to 60s, when the term *algorithmic thinking* was used at the time. The relatively recent concept ofcomputational thinking can be approached as a revival of earlier concepts combined with many of the new concepts related to the continuously changing technological apparatus that supports computational environments nowadays, as some of those considered by Wing in [Wing 2006]. Although computational thinking is broadly considered by many as an evolution or a generalization of algorithm thinking, both terms have substantial differences between them, mostly due to the existing huge gap between computational environments in the 50-70's and those available now. Algorithmic thinking has a strongbias in organizing the task of solving a problem as a procedure, which could be then coded and performed on a computer (or some machine), using a set of basic instructions borrowed from some programming language, not necessarily computational.

As commented in [Denning 2009], algorithmic thinking can be translated into the mental orientation for the formulation of problems such as conversions of some inputs into outputs, and in the search for an algorithm that can make such a conversion. It is a term mainly encompassing the organization of a process in such a way that, given as input some data, produces an output which presumably is the solution to a problem. Algorithmic thinking is closely related to and intertwined with the use of a style of computer programming named *structured programming* [Dijkstra 1968, Dahl et al. 1972, Reynolds 1981], that emphasizes the understandability (by the user) of the process described by a high-level language based computer code, usually referred to as source code.

Wing in [Wing 2006] considers computational thinking a universal set of feasible attitudes and skills, which can be adopted by individuals in general and not just by individuals linked to the computational area, for problem solving, for systems design, as well as for the understanding of human behavior. According to Wing, computational thinking (1) is supported by the mathematical way of conducting reasoning, when representing and solving problems, (2) borrows the way of thinking used in engineering, when addressing systems´ design as well as the evaluation of large complex systems, taking into account restrictions imposed by the real world (3) shares with the scientific thinking in general, aspects related to (a) computability, (b) intelligence, (c) the human mind and (d) human behavior can be addressed. Wing rephrases the definition of computational thinking by proposing a set of operational and functional definitions associated with the term, which can be briefly summarized as:

(1) to re-specify an apparently difficult problem as a similar problem that has been previously solved, eventually through the use of reductions, transformations, simulation or incorporation; (2) think recursively; (3) consider the possibility of using parallel processing; (4) treat data as code and vice versa; (5) recognize advantages and disadvantages in using aliases i.e., assigning more than one name to something; (6) recognize both the cost and the power of indirect addressing and procedure calls; (7) evaluate a program not only for its correctness and efficiency, but also for its aesthetics and a system design for its simplicity and elegance; (8) use abstraction and decomposition to address a complex problem or else to design a complex system; (9) use invariants to describe the behavior of a system succinctly and declaratively; (10) trust that a large complex system can be used and modified safely, without understanding all its details; (11) modularize something in anticipation of multiple users or perform pre-searches and caching, in anticipation of future use; (12) reflect on the problem to be solved keeping in mind prevention, protection and recovery in possible scenarios considered as 'worst case', through redundancy, damage containment and error correction; (13) consider the computational concepts of deadlocks, gridlocks and contracts as interfaces; (14) use heuristic reasoning to discover solutions; (15) plan, learn and organize in the presence of uncertainty; (16) conduct searches in spaces of possible solutions; (17) balance between time and space and between processing power and storage capacity.

The initial ideas, concepts and discussions on computational thinking were extended by the author with the addition of new considerations and arguments in [Wing 2008a], where the author goes further and discusses, among others, some conjectures about the future development of computational thinking by means of *deeper computational thinking*, based on more sophisticated abstractions. In [Wing 2008] five questions related to fundamental CS issues, taking into account the fast technological development we are experiencing, are proposed and discussed, with the goal "to stimulate deep thinking and further discussion".

Several definitions related to computational thinking found in the literature take into account different levels of abstraction and details. It is evident that the previous set of operational and functional definitions/concepts related to computational thinking involves a huge volume of knowledge related to numerous subareas of CS that, to be properly and efficiently mastered in depth, requires many years of formal study and practices, not only in the CS area, but in several subjacent areas, particularly Mathematics and Statistics. Add to that the fact that the CS area is quite volatile, mainly due to its highly technological bias, which requires constant renew of the hardware apparatus as well as the upgrading and/or downloads of new software releases. As a consequence, the user needs to constantly adjust to the new versions, as well as to revise and update the knowledge previously acquired.

Recalling that computational thinking in Wing's view is a line of thought that supposedly "can be adopted by individuals in general and not just by individuals linked to the computational area", it is very hard to foresee in which way such assumption could be concretized, as well as in which way it could be implemented, without investing many years in study and practice. Even individuals that have a strong bias towards CS can have their interests in subareas of CS that demand only the mastering of specific types of computational knowledge. The types of computational knowledge needed for turning the previous set of operational and functional definitions feasible to be mastered by an individual seems to be a very hard an endless project, taking into account, also, the high speed of technological knowledge changes. If the individual in question has its interest directed to any other branch of science, the task seems unfeasible, unless if it is done in a very superficial way, which summarizes to nothing.

On the one hand the proposed definition could be seen as an attempt to gather several relevant aspects of CS as possible subareas to be considered for skill development and practice which, eventually, will be of interest to many individuals but not to all and be tried by some, to a certain degree, due to their lack of interest/time. On the other hand, the development of the algorithmic thinking, combined with concepts of data structure and the set of basic command structures made available by procedural programming languages, should be considered as part of an individual´s education, and be carefully dosed by starting from stage (3) of Piaget's theory of cognitive development.

Acknowledgments

The authors thank UNIFACCAMP for its support.This study was financed in part by CAPES, Brazil,Finance Code 001. The authors are grateful to the reviewers for their insightful comments and suggestions.

References

[Denning 2009] Denning, P. J. (2009) The profession of IT: Beyond computational thinking, *Communications of the ACM*, v. 52, no. 6, pp. 628–630.

[Dahl et al. 1972] O-J.Dah, O. –J; Dijkstra, E. E.; Hoare, C. A. R. (1981) *Structured Programming*, Academic Press, London, 1972.

[Dijkstra 1968] Dijkstra, E. W. (1968) Letters to the editor: go to statement considered harmful, *Communications of the ACM*, v. 11, no. 3, pp. 147–148.

[Goldberg 1982] Goldenberg, E. P. (1982) Logo A Cultural Glossary, *Byte*, p. 218.

[Logo-foundation 2021] https://el.media.mit.edu/logo-foundation/

[Munari 1994] Munari, A. (1994) Jean Piaget (1896–1980), *Prospects: The Quarterly Review of Comparative Education*, Paris, UNESCO: International Bureau of Education, 31 vol. XXIV, pp. 311–327.

[Papert 1980] S. Papert (1980) *Minstorms: Children, Computers, and Powerful Ideas*, USA: Basic Books, Inc.

[Piaget 1952] Piaget, J. (1952) *The Origins of Intelligence in Children* (M. Cook, Trans.), W. W. Norton & Co. https://[redacted]/[redacted]/[redacted]-000

[Reynolds 1981] Reynolds, J. C. (1981) *The craft of Programming*, Prentice-Hall International, London.

[Wadsworth 1971] Wadsworth, B. J. (1971) *Piaget's Theory of Cognitive Development: An Introduction for Students of Psychology and Education*, New York: McKay,

[Wing 2006] Wing, J. M. (2006) Computational thinking: viewpoint, *Communications of the ACM*, v. 49, no. 3, pp. 33-35.

[Wing 2008] Wing, J. M. (2008) Five deep questions in computing, *Communications of the ACM*, v. 51, no.1, 58. doi:10.1145/1327452.1327479

[Wing 2008a] Wing, J. M. (2008) Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v.366, no. 1881, pp. 3717–3725. doi:10.1098/rsta.2008.0118

[Yadav et al. 2014] Yadav, A.; Mayfield, C.; Zhou, N.; Hambrusch, S. and Korb, J. T. 2014. Computational thinking in elementary and secondary teacher education, *ACM Transactions on Computing Education, v.* 14, no. 1, Article 5, 16 pages. doi:http://dx.doi.org/10.1145/2576872

Corresponding Author: Maria do Carmo Nicoletti, carmocrat@gmail.com