CLEI ELECTRONIC JOURNAL, VOLUME 19, NUMBER 1, PAPER 5, APRIL 2016

Philosophy of Computer Science and its Effect on Education

Towards the Construction of an Interdisciplinary Group

Sylvia da Rosa
Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300
darosa@fing.edu.uy

and

Alejandro Chmiel
Departamento de Lógica y Filosofía de la Lógica
Facultad de Humanidades y Ciencias de la Educación - UDELAR
Montevideo, Uruguay
alejandro.chmiel.gmail.com

and

Federico Gómez
Facultad de Ingeniería - UDE
Montevideo, Uruguay
fgomez@ude.edu.uy

Abstract

This article presents an interdisciplinary experience that brings together two areas of computer science; didactics and philosophy. As such, the article introduces a relatively unexplored area of research, not only in Uruguay but in the whole Latin American region. The reflection on the ontological status of computer science, its epistemic and educational problems, as well as their relationship with technology, allows us to elaborate a critical analysis of the discipline and a social perception of it as a basic science.

Keywords: Philosophy, Epistemology, Interdisciplinary, Education.

1 Introduction

Between 2009 and 2011, the Computer Science Institute of the Faculty of Engineering of the University of Uruguay (Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay) developed a series of joint activities with the Teacher Training in Informatics (Profesorado de Informática) through the Support Programme for the Teacher Training in Informatics (Programa de Apoyo al Profesorado de Informática (PAPI) (http://www.fing.edu.uy/inco/seminarios/papi/proyectos.html). The objective of this collaboration was to support and promote the academic development of the teaching staff of Teacher Training in Informatics through activities that allowed their members to take part in research groups. The activities also included the participation of teachers of Philosophy from the Faculty of Humanities and Science of Education (FHCE) based on their interest in the philosophical problems specific to the discipline of computing.

As a result of the exchange of theoretical problems on the conceptualization, formalization and computational thinking, a series of shared interests emerged between researchers from the fields of computer science education and the philosophy of computer science. One example is the link between two classical questions of philosophy of computer science: What is a program?, and; what kind of knowledge can we have of it?

The different paradigms of computer science (or informatics or computing) which may provide answers to these question have a direct impact on the didactics of the discipline. In other words, the paradigm determines the pedagogical content knowledge for computer science [1,2].

In this way, the working group established from that moment, opens a certain kind of reflection on computer science that had not been developed so far in the country, at least systematically and explicitly. It is clear that the philosophy of computer science as a discipline, with certain well-defined questions articulated in a set of reference works, has no more than thirty years, but in the last ten years the intensity of work in the area has been remarkable.

Our group developed a work proposal to the Interdisciplinary Space of University in Montevideo (Espacio interdisciplinario de la Universidad de la República) that was approved in 2012 and gave rise, from that moment, to the Interdisciplinary Center Philosophy of Computer Science (Núcleo Interdiscilinario Filosofía de la Ciencia de la Computación, (NI FCC), http://www.fing.edu.uy/grupos/nifcc) The general problem of the proposal is defined as "the absence of a culture as a basic computer science in society and rigorous theoretical reflection about technology". This absence is a factor that largely determines the partial and distorted vision that exists on computer science and technology in society and constitutes an obstacle to the educational, social and productive development. Contributing to the solution of this problem is the foundation of our project.

Many elements of informatic culture do not require a computer; in fact, historically those elements have their roots in logic and mathematics and appeared before the invention of the machines. Their influence from the beginning transcended the boundaries of these disciplines, for example the notion of algorithm or regulated procedure, although it appears originally in scientific problems, has a clear impact on thinking about thinking in a general sense.

It is through the Philosophy of Computer Science that these issues can become the object of reflection and knowledge, providing a rich source of interesting problems from both philosophical and informatic perspective. However, computers have become an omnipresence that covers almost all areas of social interaction, and technological tools are presented as transparent or unproblematic. The first consequence of this is that it reaches the general conclusion that handling a technological device is knowing informatics, which is obviously a mistake (in principle, driving a device, probably through a program, has no link with computer science concepts), bringing as a second consequence a society that perceives itself as an expert in what it is illiterate. It is worth adding that this problem is global, that is, the problems facing Uruguay in these topics can be found in any Latin American country, and even in developed countries. However, since a decade ago there is a strong and growing trend in American and European universities to create and promote integrated research institutes in various areas of knowledge. The aims are to contribute to the development of a critical vision of computing, from a reflection about its ontological status, its epistemological problems, its teaching and its links with technology. From those issues it is possible to develop a certain type of criticism that contributes, among other things, to the development of computing as a basic science in education, as is happening in many countries [3-9]. It is of fundamental importance for this critical vision, to develop some lines of work within the Philosophy of Computer Science area, as well as do research in didactics of informatics. In countries like the USA, France and United Kingdom, university academics in the area of computer science have begun to study the problem of the lack of reflection on informatics and its relationship with technology. This helps to lay the foundation for education in computer science as part of the discipline, and an interesting matter for doing research in computer science. In this regard it is worth quoting the words of Simon Peyton Jones [9] on the training of teachers for the introduction of computer science in primary and secondary school, which illustrate the treatment of the subject in computer science communities in other countries:

But who will teach the teachers? We have two main routes. First, the Computer Science departments of our universities. For the last two decades university CS departments have been utterly disengaged from school ICT, because the subject was of no interest or relevance to them. Now there is a real prospect that CS will be taught to school children, every university academics suddenly have a real stake in what is happening at school.

This paper is organized as follows: in Section 2, different ways of connecting philosophy and computer science are presented and in Section 3 the focus of our work is briefly explained. Section 4 includes some aspects of the effect of a philosophical perspective in computer science education, specially a model for doing research in didactic issues. Sections 5 and 6 present the courses and the activities aimed at society respectively and section 7 some conclusions.

2 Philosophy and Computer Science

Philosophy and computer science are deeply connected in contemporary philosophy. The degree of relevancy given to this connection may vary depending on the philosophical tradition, the approach, the field in which we are situated, and the problem we are aiming at. An undeniable fact is, however, that the link between philosophy and computer science has deepened and developed since, at least, the work of Alan Turing in [10]. Discussions on artificial intelligence are strongly related to the philosophical discussions on the notion of intelligence: the problems of computing connect directly with debates within logic, philosophy of logic and philosophy of mathematics; computational models have an enormous impact on cognitive sciences, on philosophy of mind, and even on philosophy of science. Of course, more examples could be listed.

We could establish an important distinction between two different ways of connecting philosophy and computer science. On one hand, (a) when concepts coming from computer science are used as an input for certain philosophical reflections, as it is the case in philosophy of mind and philosophy of mathematics. Colburn [11] provides a general discussion on the relationship between philosophy and computer science. On the other hand, (b) when concepts belonging to this discipline are seen from a *metatheoretical* point of view with the intention of elucidating aspects of these concepts that fall out of the disciplinary discussion even when this aspects are relevant to the ontological and epistemological status of the discipline. In this sense, a list of philosophical problems in computer science can be revised in [12] and [13].

A third way of addressing the issue could be added (c) referred to the historical studies in computer science, in this case we are necessarily in need of elements that belong to philosophy of science. Carrying out research in problems of philosophy of computer science, in any of the ways described above, requires interdisciplinary research groups. Our team has focused mainly on some of the issues that can be addressed from perspective (b) above, as we will explain briefly in next section.

3 Philosophy of Computer Science

Philosophy of computer science, understood in this sense, takes some concepts that both belong to the discipline and are at the base of the ontological and epistemological discussions of that discipline. Our research group has been working mainly around four axes that shape the discussion. Firstly, because the answer given to some fundamental questions will define the way in which we understand computer science. Secondly, because these answers are related to each other, given their conformation as a consistent theoretical position. The problems we are referring to could be listed as follows:

- The ontological problem: what is a computer program? [11, 14].
- The implementation problem: what is the link between an abstract algorithm and the machine that implements it? [15, 16].
- The problem of correctness: what does it mean for a computer program to be correct? [17].
- The computing notion: what is computing? [18], the limits of computability and the interpretation of the Turing-Church thesis, the link between the programs and the machines that compute those programs [19].

An excellent summary of these problems can be found in [13]. In this paper we will only mention some aspects of the *ontological problem* and the connections that could be made with the *problem of implementation* and *the problem of correctness*. The ontological problem globally affects our way of picturing computer science and, more specifically, it makes it possible for us to discuss in which sense computer science is, or is not, a formal science.

3.1 The ontological problem

The ontological problem is fundamentally based on the question “what is a computer program?” The possible answer tends to be framed in the abstract/concrete duality, it tends to be answered via an argumentation that reduces the notion of software to one of these poles or, via an argumentation against the possibility of such a reduction. We could defend that a computer program is an abstract entity, a mathematical entity, with independence of its physical implementation, or we could sustain that a program is a concrete entity, a physical entity, this is: the physical machine that actually computes that which the program develops.

The distinction might not have an ontological relevancy, as Moor claims in [14], as long as, as he sustains, the limit between software and hardware is contextual, and varies from case to case. For instance, something that is implemented by the hardware itself in one case, has to be programmed as a software in other cases.

Nevertheless, as later authors have noticed, particularly Colburn in [11], the dualist distinction cannot be reduced. An argumentation in this sense states as follows: if we consider that without a linguistic description of the program, this is, only with a physical implementation, we are left with no normative criterion for correctness [20], then we will have to conclude that the notion of program cannot be reduced to its concrete counterpart, unless we accept the possibility of being deprived of a criterion for correctness. In other words, if we observe the execution of a physical machine, or better, if we observe the actual computing it performs and we do not have any supplementary information, then we cannot decide whether the program is working correctly or not. This could be seen as a particular case of a broader perspective, related to the "rule following" problem [21, 22].

Therefore, when it comes to ontologically determine what a computer program is, the abstract/concrete tension cannot be reduced to its concrete pole, given the aforementioned reasons. Another possibility opens: that of reducing the tension to its abstract pole. In this case, we would say that a computer program is an abstract entity even when not implemented and, what is more, even when it is not possible to implement it. In this sense, we would say that computer science is a field within mathematics, and that would result in the exclusion from the discipline of some areas that are nowadays considered to be a part of computer science. This result would not, at first glance, do justice to the current practices in computer science.

As Eden correctly remarks in [23], the ontological discussion on software divides the theoretical discussions into three paradigms which connect ontology with a particular epistemological and methodological notion about computer science:

1. Rationalist paradigm: computer programs are *abstract* entities, computer science is a field within mathematics and, the notion of correctness of a computer program, is founded on *a priori*, purely deductive, procedures.
2. Technocratic paradigm: computer programs are *concrete* entities, computer science is a field within engineering and, the notion of correctness of a computer program, is founded on *a posteriori* procedures that are generally related to testing.
3. Scientific paradigm: computer programs are entities that cannot be reduced neither to concrete nor to abstract entities and, in this sense, computer science is a part of natural sciences; the adequate methods to test correctness combine *a priori* and *a posteriori* procedures.

4 Genetic Epistemology and Computer Science Education

Computer Science Education (or didactics of informatics) raises the study of educational problems of computing described in the pedagogical content knowledge mainly by the questions *what, why, who* and *how* to teach the discipline [1,2]. Finding answers to these questions concerns academic and computer scientists dedicated to the study of education in the discipline [6,8,9], related to psychological and epistemological themes. Research results can support the elaboration of patterns teaching for classroom practice, based on solid foundations.

Our research focuses mainly on the *how*, leading the question from how to teach towards how students learn. Our epistemological position on the construction of knowledge is the theory Genetic Epistemology of Jean Piaget [26-32]. The theory is concerned with studying the processes and mechanisms by which the human mind passes from one state of certain level of knowledge to another state of greater level of knowledge. The notions of "minor" and "major" are inherent in the nature of the kind of knowledge of each discipline (mathematics, physics, biology, etc.) that determines whether a state of knowledge is higher or lower than other in terms of validity. Genetic Epistemology is concerned with explaining the processes of construction of such knowledge, independently of their specific nature [26], and establishes that there is no discontinuity in that process and that the mechanisms involved are the same at all stages of such construction [44]. Hence, it provides a model applicable to the study of the construction of knowledge in all fields and at all stages of development, which we use to investigate about the construction of knowledge about data structures and basic algorithms by novice students.

The empirical data that nourish genetic epistemology come from two sources. On the one hand, from the experiments conducted by Jean Piaget throughout his life, concerning the development of the most basic levels of knowledge (from birth to adolescence). These studies gave rise to genetic psychology, which is the best-known part of the work of Piaget and has had influence on different theories, such as cognitive psychology, constructivism, the theory of mental models and Neo-Piagetian theory. On the other hand, from the critical historical analysis that Jean Piaget and Rolando Garcia used to study the evolution of scientific theories as empirical material provided from the more formalized levels of scientific knowledge. Taking into account processes inherent in any construction of

knowledge, the authors developed an explanatory model based on a general process called the intra-inter-trans triad, present in both the genesis of the thought of the epistemic subject and in the sociogenesis of scientific thought [32]. This model represents the largest contribution of Genetic Epistemology, which remains open.

Our group has conducted several investigations, trying to build an instance of the mentioned model, for the study of the construction of concepts of computer science from their psychogenesis until their formalization as school theme. One of the principles of the theory is that the source of knowledge is given by the subject's interaction with the environment. In a similar way as Brousseau's theory of didactical situations [33,34], which considers the environment as mathematical problems that the student faces, we also start from situations in which the student must resolve instances of basic algorithmic problems (for example, sort or count objects, search for items in a list). The student is capable of solving the problem in action, that is to say, he/she knows to do it, but he/she is unaware of how he/she does. Our first goal is to help the student to transform what is done in action into a correct natural language description of the algorithms involved, as a first step towards conceptualization. This constitutes to some degree, a didactic situation in Brousseau's theory. Our theoretical basis is the general law of cognition [27,30], which we briefly explain in the following.

In Piaget's theory, human knowledge is considered essentially active, that is, knowing means acting on objects and reality, and constructing a system of transformations that can be carried out on or with them [26]. The more general problem of the whole epistemic development lies in determining the role of experience and operational structures of the individual in the development of knowledge, and in examining the instruments by which knowledge has been acquired ***before*** the formalization. This problem was deeply studied by Piaget in his experiments about genetic psychology. From these he formulated a *general law of cognition* [27,30], governing the relationship between know-how and conceptualization and represented it by the following diagram

$$C \leftarrow P \rightarrow C'$$

where P represents the periphery, that is to say, the more immediate and exterior reaction of the subject confronting the objects, to solve a problem or perform a task. This reaction is associated to pursuing a goal and achieving results, without awareness neither of actions nor of the reasons for success or failure. The arrows represent the internal mechanism of the thinking process, by which the subject becomes aware both of the coordination of his/her actions (C in the diagram), and of the modifications imposed to objects, as well as of their intrinsic properties (C' in the diagram). The process of the grasp of consciousness described by the general law of cognition constitutes a first step towards the construction of concepts.

Piaget also describes the cognitive instrument enabling these processes, which he calls the *reflective abstraction* and *constructive generalization*. Reflective abstraction is described as a two-fold process [27]: in the first place, it is a projection (transposition) to the plane of thought of the relations established in the plane of actions. Second, it is a reconstruction of these relations in the plane of thought adding a new element: the understanding of conditions and motivations.

The motor of this process is called by Piaget the search of reasons of success (or failure). On the other hand, facing new problems presenting variations and similarities with the old ones causes a desequilibrium of students' cognitive structures which have to be transformed in order to attain a new equilibrium, making possible the construction of appropriate knowledge to solve the new situation. Once a particular method is understood, students' reasoning attempts to generalize what has been successfully constructed to all the situations, by means of inductive generalization where deductions or predictions are extracted from observations of the new objects. A process of inferences and reflections about the subject's actions or operations by means of constructive generalization gives raise to new methods [28,29] and opens possibilities for constructing structures.

4.1 Our model for computer science education

In order to illustrate the meaning of the theoretical principles into our model, we use one of our studies: the case of sorting algorithms. In it, the students were required to pick up numbered cards from a bag and construct an ordered row on the table. Once done, they were asked to explain in natural language how they did and why they succeeded, and finally to write down their descriptions in a pseudo-code previously introduced [42].

Our model is a synthesis of investigations about the passage from levels of knowledge that we have classified as follows:

- Instrumental knowledge or knowledge in the plane of actions: this is the knowledge used by students to *know how* to solve instances of problems, for example how to sort a given set of numbered cards. They follow an objective, and verify their results but they are unaware of their actions or of the reasons of their success or failure (P in the diagram). This level corresponds to the intra stage of Piaget's theory.

- Conceptual knowledge or conceptualization: this is the knowledge used by students to detach from particular situations and begin to elaborate general methods in order to solve general problems. For example, they can provide accurate descriptions in natural language of the method used to reach an ordered row of numbered cards. Material actions are transformed into actions in the plane of thought (concepts) by a process in which subjects become aware of the coordination of their actions. For example, they become aware that they compare a card to be inserted with those in the row *until* some condition is satisfied, (C in the diagram above), and of the modifications of objects (for example, they realize that the number of cards in the bag diminishes until the bag becomes empty, C' in the diagram above). The questions that act as the motor of the process are related to the reasons of their success (or failure). This level corresponds to the inter stage of Piaget's theory.

- Thematized knowledge or formalization: this is the knowledge used by students to formulate in some formal language introduced by the teacher, their constructed concepts. For example, they implement a program of sorting algorithm in a programming language, from their descriptions. A new cycle of the triad occurs to construct knowledge of the new objects: students interact with the elements of the formal language, following the law of cognition. In this interaction, a synthesis of the trans stage which coordinates C and C' takes place giving rise to new mental structures, open to new possibilities.

Regarding the methodology of our research, the passage from intra to inter stage is investigated by means of conducting individual interviews, in the sense of Piaget's studies of genetic psychology [37-41]. The passage to the trans stage is investigated by means of conducting instructional episodes where students work in groups and some formalism is introduced (mathematical language, pseudo code and/or programming languages). In this part, our methodology follows Piaget's studies about the role of social relations and formal education in knowledge construction [45,46] (Piaget's ideas about social construction are integral to his epistemological theory but less known than those about child's construction of logical thought.). The goal is to help students in establishing correspondences between the concepts that they have previously constructed and expressions of the formalism in order to obtain formal descriptions of their solutions. Our investigations have been conducted with students entering university or enrolled in the final year pre-university. That means that they have no (or very little) experience with programming (in Uruguay Computer Science is not part of High School curriculum). All the research episodes were recorded and/or filmed and students wrote out some of their responses.

4.2 The effect of the ontological problem on computer science education

The problems inherent to the definition of the ontological status of the notion of *computing* and the limits of its scope are the basis of philosophical debates about the relationship between physical and abstract computing, the dual nature of computer programs, the eventual bifurcation of the Church-Turing thesis into abstract and concrete systems and even methodological aspects as the relationship between formal verification and testing program [47].

These problems have an effect on investigations about the construction of knowledge of basic algorithms and data structures, because the goal is to gather information to help students to learn programming, not just to write program texts. As programmers it is expected that they can solve problems, choose the best solution, construct a program that can be executed in a computer, analyze eventual errors and correct them. Those points imply to know about the way the data and the code are represented and organized inside the computer.

The ontological problem raises the questions: what kind of knowledge do students construct about the non-textual nature of a program? If we consider that a program is in some sense a synthesis between a text and a machine that executes it, knowledge about the text becomes necessary but not sufficient knowledge to deal with programming problems. How can we learn on the knowledge of the students about execution aspects? These questions are the matter of further work. However, we have discussed some ideas, the main of which consists in applying the general law of cognition to make students aware of operations related to program execution, that are unconscious. Once a program text is written, for executing it on a machine, a correspondence between formal and effective parameters takes place. Reflecting about how such relationship can be established, before any formalization, is a way of discovering how students think.

5 Teaching activities

Our group has among its objectives to contribute to the training of teachers and students of computer science providing insight into philosophical problems, especially in the field of education. Undoubtedly, computer teachers should be experts in their course content, but they also need to know with some depth, significant aspects of the discipline that allow them to expand their perspectives on the field and therefore improve the quality of instruction [43]. Two main courses are taught (information about the courses is available in http://www.fing.edu.uy/grupos/nifcc/cursos.html):

- *Introduction to Computer Science Philosophy* is a course about the problems of philosophy of computing, aimed at academics, students and teachers of computer science and philosophy. In the course, the problems described in Section 3 are presented and discussed, based on an extensive bibliography. Especially the work of Matti Tedre [48,49], in the form of "Lecture Notes in The Philosophy of Computer Science" (available in cs.joensuu.fi/~mmeri/teaching/2007/philcs/) has been a source of ideas and a model to follow in teaching the course.

- The course *Genetic Epistemology and applications to Computer Science Education* is issued each year since 2010. It is aimed at teachers of computer science in general and especially those intending to engage in research in computing education. In the course the theoretical framework is introduced, as well as the model for research about knowledge on concepts of basic algorithms and data structures, including examples of empirical studies.

6 Activities aimed at society

Our proposal has also a goal of social perspective which seeks clarify public opinion about the status of computing, trying to dismantle the false notion that assimilates the science of computing to the management or the ability to use technology (or some devices) [24,25]. This notion arose given the ICT-dependent context in which we find ourselves and the lack of a proper computer science education since primary school. In this sense we agree with thoughts and actions that are carried out in other countries, as illustrated by the following words of Simon Peyton Jones [9]:

> *But that was the easy part! Now the ground war begins: school by school, head teacher by head teacher, we must make the case, convey the vision, offer support and teaching materials, and train teachers. Explain what computer science is. We need to find ways to explain what our discipline is, in ways that make sense to parents, civil servants, and politicians, not just to the technical community. Clearly distinguishing disciplines from skills and technologies is helpful.*

Our group holds regular outreach activities aimed at teachers, students and education authorities, as well as academics from computer science, covering issues such as:

- Use and abuse of the term computing
- Benefits of studying computer science, in general education of the citizen
- Computer science education in formal education
- How informatics affects the teaching of other disciplines
- Distinction between computer science and its applications

Through the activities, the following meaning of terms are discussed and clarified:

- computer science: a science with the same general characteristics and problems of definition that other disciplines and sciences, such as mathematics, language, or physical. The objects of studying of computer science are data and mechanisms to its management,

- didactics of computer science: like other disciplines (such as mathematics), the computing has a field that is its teaching, that is, the knowledge related to the teaching and learning of computing concepts,

- technology education: is what is called "technological literacy"

- technology in education: it is understood as the knowledge to integrate and coordinate technological tools and services to teaching practices, to support the teaching and learning of any discipline.

The last point is particularly important because it highlights the need for the education system of directing and developing the pedagogical training of teachers so that they are able to use technology as an enhancer of educational practices, regardless of the discipline. Pedagogy and technology have always been linked. The rise of digital technologies on society in recent years has made this relationship more explicit, because of the abundance of technology products and services in all activities of society and especially in education.

That abundance has not been accompanied with the pedagogical coordination and integration of resources and tools of technology in the educational activity. The risk is that, on the one hand, the instrumental and thoughtless use of technology displaces the pedagogy of the conductor role of the teaching-learning process and on the other hand, it is installed the mistaken belief that didactics of informatics must deal with teaching and learning to use computer products or services. Reduce (or avoid) this risk implies a responsibility and a commitment of the education system. For this, the pedagogical training of teachers should be transformed so as to include the study and research on the implications of technological factor on pedagogy.

Among the activities aimed at society it is worth mentioning the editions of the Meeting of Education in Computer Science (Encuentro de Educación en Ciencia de la Computación (EECC)), carried out yearly from 2012, with guests from the University of Buenos Aires (UBA) and the University of Cordoba (Argentina) respectively. Especially in the latter University there is a group that works on philosophy of computer science with which we maintain contact and hope to expand academic collaboration. It is also worth mentioning the panel discussion organized by our group under the Iberoamerican Congress on Higher Education in Computing (Congreso Iberoamericano de Educación Superior en Computación (CIESC)) in Montevideo in 2014, with the participation of researchers from Argentina, Paraguay and Uruguay, on the situation in the region of the relationship between higher education and secondary education in computing.

7 Conclusions

The philosophical reflection on computing makes a fundamental contribution to computer science in at least three areas:

1. Clarifies metatheoretical notions that the discipline does not care to clarify, by the very nature of such issues.

2. Achieves building a critical point of view about the discipline, as a fundamental contribution for interdisciplinary dialogue, including didactic and pedagogical reflections. This way of linking the discipline to several social levels, as the socio-economic and cultural systems, is of great importance given the current situation in which computer science is confused with the use of technologies.

3. Provides an opportunity for teachers in computer science as well as scientists and engineers, to have a first approach to the history and philosophy of computing and its conceptual connection to other areas of human knowledge.

Our research group is beginning to build an area of study of these problems.

References

[1] M. Saeli, J. Perrenet, W. Jochems and B. Zwaneveld, *Teaching Programming for Secondary School: a Pedagogical Content Knowledge Perspective.* http://www.mii.lt/informatics_in_education/pdf/INFE177.pdf
[2] P. Hubwieser, “Towards a Conceptualization of Pedagogical Content Knowledge for Computer Science”, *ACM 978-1-4503-2243-0/13/08 (ICER 2013),* 2013.

[3] P. Bradshaw and J. Woollard, *Computing at school: An Emergent Community of Practice for a Re-emergent Subject,* http://www.icicte.org/Proceedings2012/Papers/15-2-Bradshaw.pdf
[4] Technical report of The Computer-Science-Teachers-Association website. *The New Educational Imperative: Improving High School Computer Science-Education,* http://csta.acm.org/Communications/sub/DocsPresentationFiles/White_Paper07_06.pdf
[5] The Association-Enseignement-Public-&-Informatique-(EPI), website: http://www.epi.asso.fr/
[6] G. Dowek and coll, *L'enseignement de l'informatique en France, il est urgent de ne plus attendre.* http://www.epi.asso.fr/revue/articles/a1312b.htm
[7] The Computer Science Teachers Association (csta) website: http://csta.acm.org/Curriculum/sub/K12Standards.html
[8] Computing in the national curriculum. A guide for primary teachers. http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf
[9] S. Peyton Jones, *Computing at school in the UK.* http://research.microsoft.com/en-us/um/people/simonpj/papers/cas/ComputingAtSchoolCACM.pdf
[10] A.Turing, "On Computable Numbers, with an Application to the *Entscheidungsproblem*", Proceedings of the London Mathematical Society, Series 2, 42, pp. 230–65, 1936.
[11] T. Colburn, Philosophy and Computer Science. Armonk, N.Y., M.E. Sharp, 2000.
[12] R.Turner and A. Eden, "The Philosophy of Computer Science", Stanford Encyclopedia of Philosophy, Edward N. Zalta (ed.), 2011. http://plato.stanford.edu/archives/win2011/entries/computer-science/
[13] R.Turner, "The Philosophy of Computer Science", The Stanford Encyclopedia of Philosophy, Edward N. Zalta (ed.), 2014. http://plato.stanford.edu/entries/computer-science/
[14] J.H.Moor, "Three Myths of Computer Science", The British Journal for the Philosophy of Science, 29(3), pp.213–222, 1978.
[15] W.J. Rapaport, "Implementation Is Semantic Interpretation", The Monist, 82(1), pp.109–130, 1999.
[16] W. J. Rapaport, "Implementation as Semantic Interpretation: Further Thoughts", Journal of Experimental & Theoretical Artificial Intelligence, 17(4), pp.385–417, 2005.
[17] J. H. Fetzer, "Program Verification: The Very Idea", Communications of the ACM, 31(9), pp.1048–1063, 1988.
[18] B. J. Copeland, "What is Computation?", *Synthese*, 108(3), pp.335–359, 1996.
[19] D. Chalmers, "Does a Rock Implement Every Finite-State Automaton?", Synthese (108), pp.309–333, 1996.
[20] P. Kroes, Technical Artefacts: Creations of Mind and Matter: A Philosophy of Engineering Design. Dordrecht, Springer, 2012.
[21] L. Wittgenstein, *Investigaciones filosóficas*, México: UNAM, 1988.
[22] S. Kripke, Wittgenstein on Rules and Private Language. Cambridge, MA: Harvard University Press, 1982.
[23] A. Eden, "Three Paradigms in Computer Science", Minds and Machines 17(2), pp.135–167, 2007.
[24] R. Gamboa and R. Page, "How Computers Work: Computational Thinking for Everyone", in *Proc. First International Workshop on Trends in Functional Programming in Education (EPTCS 106)*, http://eptcs.web.cse.unsw.edu.au/content.cgi?TFPIE2012
[25] The Royal Society website. Computing in Schools, Shut down or restart? http://royalsociety.org/education/policy/computing-in-schools/
[26] J. Piaget, *Genetic Epistemology,* (a series of lectures delivered by Piaget at Columbia University, translated by Eleanor Duckworth), https://www.marxists.org/reference/subject/philosophy/works/fr/piaget.htm
[27] J. Piaget, *La prise de conscience.* France: Presses Universitaires de France, 1964.
[28] J. Piaget, *L'equilibration des Structures Cognitives, Probleme Central du Developpement*. France: Presses Universitaires de France, 1975 .
[29] J. Piaget, *Recherches sur la Generalisation.* France: Presses Universitaires de France, 1978a.
[30] J. Piaget, *Success and Understanding*. Harvard: Harvard University Press, . 1978b
[31] J. Piaget and coll., *La Formation des Raisonnements Recurrentiels.* France: Presses Universitaires de France, 1963.
[32] J. Piaget and R. García, *Psychogenesis and the History of Sciences.* New York: Columbia University Press, 1980.
[33] G. Brousseau, *Theory of didactical situations in mathematics.* Dordrecht: Kluwer, 1997.
[34] P. Sadovsky, *La Teoría de Situaciones Didácticas: un marco para pensar y actuar la enseñanza de la Matemática*, 2004. http://www.buenosaires.gob.ar/areas/educacion/cepa/teoria situaciones.pdf
[35] S. da Rosa and F. Gómez, An educational methodology based on the work of students. *The Clei Electronical Journal website,* vol. 13, Nr. 2. http://www.clei.cl/cleiej/volume.php
[36] S. da Rosa, Designing Algorithms in High School Mathematics. Lecture Notes in Computer Science, vol. 3294, Springer-Verlag, 2004.

[37] S. da Rosa, The learning of recursive algorithms and their functional formalization, 2005 https://www.fing.edu.uy/~darosa
[38] S. da Rosa, "The Learning of Recursive Algorithms from a Psychogenetic Perspective", in *Proc. 19th Annual Psychology of Programming Interest Group Workshop, (PPIG2007),* Joensuu, Finland, Jul 2007.
[39] S. da Rosa, "The Construction of the Concept of Binary Search Algorithm", *in Proc. 22th Annual Psychology of Programming Interest Group Workshop, (PPIG2010)*, Madrid, Spain, Oct 2010.
[40] S. da Rosa and A. Chmiel, "A Study about Students' Knowledge of Inductive Structures", *in Proc. 24th Annual Psychology of Programming Interest Group Workshop, (PPIG2012),* London, United Kingdom, Nov 2012.
[41] S. da Rosa, "Psychogenesis of the Concept of Recursion", *in Proc. Congreso Iberoamericano de Educación Superior en Computación, (CIESC2003),* 2003.
[42] S. da Rosa, "The construction of knowledge of basic algorithms and data structures by novice learners", in Proc. 26th Annual Psychology of Programming Interest Group Workshop (PPIG2015), Bournemouth, United Kingdom, Jul 2015.
[43] Judith Gal-Ezer and David Harel. *What (Else) Should CS Educators Know?* http://www.openu.ac.il/Personal_sites/download/galezer/what-else.pdf
[44] J. Flavell, *La psicología evolutiva de J. Piaget*, Buenos Aires: Paidós, 1968.
[45] J. Youniss and W. Damon, *Social construction in Piaget´s Theory*. In H. Beilin and P. Pufall (comp.), Piaget´s Theory: Prospects and possibilities. Hillsdale, N.J.: Erlbaum, 267-285.,1997.
[46] E. Ferreiro, *The acquisition of cultural objects: the case of written language*, Prospects, Vol 26, Issue 1, 131-140, 1996.
[47] J. Blanco, P. García and R. Cherini, *Convergencias y divergencias en la noción de computación. Rev. iberoam. cienc. tecnol. soc.* 2012, vol.7, n.19 pp. 111-121. ISSN 1850-0013, http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1850-00132012000200008&lng=es&nrm=iso
[48] M. Tedre, *The Science of Computing: Shaping a Discipline,* 2014, CRC Press, ISBN 9781482217698.
[49] M. Tedre, *The Development of Computer Science – A Sociocultural Perspective*, 2006, University of Joensuu, ISBN 952-458-866-8.