Computer Science – A Language of Technology

Avi Cohen[1] and Bruria Haberman[2]
[1] *The Ministry of Education, Israel*
Avi@CSIT.org.il
[2] *Computer Science Dept.*
Holon Institute of Technology, and
Davidson Institute of Science Education
The Weizmann Institute of Science, Rehovot, Israel
bruria.haberman@weizmann.ac.il

Abstract

The field of computer science has been rapidly developing since its recognition as a stand-alone discipline. The dynamics of the field led to its inadequate public image and posed challenges regarding how to make computer science studies more appealing to students. Recently, computer science has been recognized as a language of natural sciences, and its synergy with these sciences became noteworthy. This paper illuminates another facet of computer science. We call for the acknowledgement of computer science as a *scientific paradigm, which is a language of technology*. The language describes structures, processes, relationships, and communications. We believe that this view expands the responsibility of computer science in the contemporary world and legitimates its status as a basic language that is essential for acquiring scientific and technological literacy.

1. Introduction

Educators agree that it is important to ensure that high-school graduates will be ready to meet the demands of tomorrow's high-tech society. Recently, the importance of all students becoming "computer savvy, not merely computer literate" has become increasingly recognized [11]. The computer science education community have stated that computer science provides the knowledge and skills foundation for contemporary technological advances. For example: "Maintaining our ability to meet present and future challenges requires us to acknowledge computer science as a core element of all STEM (science, technology, engineering, and mathematics) initiatives" [14, p. 15]. Strengthening the status of computer science as a full-fledged and self-contained subject in the educational system is most important, with expectations that (like mathematics) it should be taught as an essential core subject. However, this mission is not yet simple to achieve, probably due to the field's public image. The increasing complexity of the field led to an inadequate external image and poses new challenges in motivating students pursuing computer science as a career choice or a course of study in which to major [4,9,14].

In the recent years, enrollments in computer science undergraduate programs have been dropping, a phenomenon that has raised great concern not only in academia, but in the ICT industry and among policy makers [10]. There are many factors that have contributed to the decline in student interest, some of which relate to the lack of understanding the essence of computing and its importance in the rapidly developing high-tech world [12]. Moreover, students have not yet recognized the benefits of studying the subject, as it may serve as an entry point to other disciplines: "one can major in computer science and do anything" [17]. Recently, initiatives to reduce the declining enrollments have been undertaken. For example, the Computer Science Teachers Association has worked tirelessly to promote computing education in high schools [12,14]; the ACM Education Board has developed an informational brochure to counter the lack of awareness of opportunities in the computing field. In addition, institutions have pioneered pedagogical strategies aimed at increasing the attractiveness and relevance of computing to a wide range of students [7,11]. However, "the problem nonetheless persists" [12]; the status of the subject in the educational system is still underestimated and computer science has not yet been recognized by the educational system as a whole as a subject to be studied that provides the knowledge and skills foundation for contemporary technological advances. Addressing the negative image of the computing profession still poses a great challenge [10] and the need for a simple and attractive description of its essence is obvious. In this paper we suggest how to address this problem.

2. Computer Science – A Language of Technology

2.1. The need to illuminate computer science

Along with the rapid development of computer science, many efforts have been made to make it more understandable and approachable, for insiders, prospective students, and outsiders (especially stakeholders and policy makers) [3,4,14].

The authors of this paper have personally experienced difficulties in convincing policy makers in the K-12 education system about the increasing importance of computer science as part of the required knowledge base of every citizen in a technological and global society. These difficulties probably arise owing to an inadequate public image of the field: "In the face of confusing definitions of computer literacy, information fluency, … many schools have lost sight of the fact that computer science is a scientific discipline and not a “technology” that simply supports learning in other curriculum areas" [14, p. 14]. Specifically, during the last year, the first author (who recently has served as the head of the computer science department in the Israeli Ministry of Education) encountered a most problematic situation owing to deep budget cutbacks; as a result, it become crucial to justify to policy makers why studying computer science is essential. As part of searching for convincing arguments, he performed face-to-face interviews with computing professionals (i.e., high-school teachers, faculty in academia, and practitioners in high-tech industry). Because of the inconsistent self-image of computer science [14], it was not surprising that the study subjects stated different descriptions of the domain. Some were related to skills, critical thinking and problem-solving challenges characteristic of computer science; others described computer science as a branch of applied mathematics; others chose to refer to expertise related to core technologies and applications. Most focused on algorithmics as the main issue of computer science. The dichotomous description of computer science in terms of (a) characteristic problem-solving skills versus (b) core technologies and applications motivated us to seek a simple description that bonds both aspects. Following the statement that computer science is a scientific discipline [5,14], we agree that the misconception *computer science equals technology* should be diminished and eventually discarded. However, we strongly recommend that the connection of computer science to contemporary technology should be illuminated. Specifically, it is important that the students be better educated regarding the long-standing fundamental principles of the discipline (emphasizing that they are above specific technologies) while illustrating linkages between contemporary computing and its applications.

Based on these considerations, we call for the acknowledgement of computer science as a scientific paradigm, which is a language of technology. We believe that this view expands the responsibility of computer science in the modern world and legitimates studying computer science as a "basic language" in the educational system, starting at the K-12 level and proceeding to scientific and engineering faculties. Moreover, we believe that such a description of computer science to prospective students, educators, and policy makers, can contribute to efforts to revitalize interest in computing programs and in computing itself as a career path.

2.2. Is computer science a language?

The concept of a language is not so simple to define: "As with any complex, emergent concept, language is somewhat resistant to definition; however, most would agree that language is a system of communication or reasoning using representation along with metaphor and some manner of logical grammar. Many languages use gestures, sounds, symbols, or words, and aim at communicating concepts, ideas, meanings, and thoughts, though the problem of linguistic vagueness often rears its head when we try to distinguish between these things." [http://en.mcfly.org/Language accessed: May 6 2007].

Does computer science have the characteristics common to a language? The answer follows from its original description: "The discipline of computing is the systematic study of algorithmic processes that describe and transform information" [3, p.12]. Actually, the discipline provides a means of expressiveness, reasoning, and communicating by use of conventional symbols that can be employed to describe processes, structures, relationships and communications.

Computer science is considered to be closely related to mathematics. The first look at scientific language is taken from mathematics: "Mathematics is pure language - the language of science. It is unique among languages in its ability to provide precise expression for every thought or concept that can be formulated in its terms" [1, p. 435]. To expand on the concept of computer science as a language, we can look at the common characteristics of a language within computer science itself. Like any other field, computer science has its own brand of conceptual and technical terminology. In some cases, a word or concept in general usage has a different and specific meaning within computer science; in other cases, unique terms and statements have been created that do not exist outside of computer science. The rapid growth of the field, along with the emergence of new technologies that have spread into new application areas [4] established the basis for a dynamic taxonomy of concepts, as well as modelling tools. It is perhaps too optimistic to believe that this description will pass without criticism. It may take further evolution of the computer science field to build, like in mathematics a generally accepted taxonomy (e.g., http://people.uncw.edu/hermanr/MathTax/ accessed: May 13, 2007).

Computers and their accompaning formalisms "complement the already wide-spread use of mathematical and logical formalisms" [15, p. 21]. Computer science is constantly contributing to other fields by demonstrating how to model their processes as information processes [5]. Actually computer science is "poised to become as fundamental to science, and in particular the natural sciences, as mathematics has become to science, and in particular the physical sciences" [16, p. 26]; "Its concepts and theorems are starting to prove fundamental in explaining natural and physical phenomena." [16, p. 73]. Furthermore, computer science concepts provide levels of abstraction "allowing scientists from different fields to understand and learn from each other’s solutions, and ultimately for scientists to acquire a set of widely applicable complex problem solving capabilities, based on the use of a generic computational environment, in the same way that they learn universally applicable mathematical skills" [16, p. 8]. Apparently, these arguments facilitate recognizing computer science as a scientific language.

2.3. A language of what?

To answer this question, one has to delve into history and to examine the milestones that created contemporary computer science as we know it. Interestingly, even though algorithms have been known since the ninth century, the need to define computer science as a stand-alone discipline arose only in the early 1940s, with the invention of the first electronic computers. Aapparently, this matched timing of events has clarified the essential linkage between computer science and technology and implies that technology reached the stage where it needed a language that describes it. In spite of Dijkstra's famous statement: *Computer science is no more about computers than astronomy is about telescopes*, one cannot ignore that the term "Computer Science" evolved from the technology that created the need.

Why computer science is a language of technology? The answer follows from the notion of the concept *technology*. Technology is the need to solve the practical problems of the intelligent man [2]. It is the application of scientific and other knowledge, tools, and skills to solve practical problems and extend human capabilities [9, 13]. "Technology is best described as a process, but it is more commonly known by its products and their effects on society… Technology's role is doing, making and implementing things. The principles of science, whether discovered or not, underlie technology" [9, p. 1].

Computer science has three origins: math, engineering, and science [3] and therefore its fundamental principles and modelling tools are built upon these components. Recently it has become obvious that "Conceptual and technological tools developed within computer science are … starting to have wide-ranging applications outside the subject in which they originated, especially in sciences investigating complex systems" [16, p. 8].

The *Science 2020* report refers to the status of contemporary computer science: "Computer science teaching and research is currently at an awkward crossroads where it needs to decide whether it is something that serves other disciplines, is an engineering exercise, or a real science in its own right". The report clarifes that computer science is a science in its own right; however, "clearly, there are significant aspects of computer science that are purely engineering" [16, p. 73]. The report uses technology-oriented arguments to explain how computing is essential to other sciences as well as to addressing information technology issues (e.g., managing the data explosion). Integration of computing, computer science and mathematics leads to integration of modelling, experimental and theoretical approaches in science. In particular, "Scientific computing platforms and infrastructure are making possible new kinds of experiments that would have been impossible to conduct only 10 years ago, changing the way scientist do science." [16, p. 14].

Over the years, publications in computing often literally used the concept *technology* (e.g., "core technologies of computing") or related to aspects of solving practical problems. For example, Smith stated that computer science should be used as "a testbend for technology to enable the computerization of may areas of human knowledge" [15, p. 23]. Denning's "great principles" framework, for example, tries to balance between scientific concepts and practices, and relates to computing mechanics and design, each of which is expressed in every computing core technology in its own way [4]. Apparently, the framework illuminates various aspects of "solving practical problems" and refers to cross-fertilization between computer science and technology: "It is best to think of practices and principles in an endless cycle of mutual reinforcement; our practice is shaped by principles, and our perceptions of principles are shaped by practices" [4, p. 339].

Obviously, besides being a science in its own right, computer science serves as a platform for applying scientific and other knowledge to practical tasks, a designation that is compatible with the notion of technology. We believe that these arguments are a basis for recognizing computer science as a language of technology. The language describes structures, processes, relationships, and communications. It supports abstraction, formalization, and knowledge representation. Since computational concepts are deeply embedded into everyday thinking in many fields [17], the language facilitates "doing and understanding" technology.

Why we should not be afraid of relating computer science to technology? Computer science has been suffering throughout the years from an underestimated self-image, and has fought to be recognized as a science. This situation, along with trying to diminish the common misconception *"computer science equals technology"*, explains the reluctance to relate computer science to technology. However, nowadays, recognition of computer science as "a real science in its own right" is becoming a fact [5,6,16]: "Computer science studies information processes both artificial and natural. It helps other fields study theirs too." [5, p. 28]. In spite of the statement that "the old definition of computer science – the study of phenomena surrounding computers – is now obsolete" [6, p. 14], one cannot ignore that advanced computing platforms and infrastructure (which obviously are technological artifacts) are making it possible to change the way scientists do science. Furthermore, "Towards 2020, … dramatic in its impact, will be the integration of new *conceptual and technological tools* from computer science into the sciences " [16, p. 8].

Recognition of computer science as a science should encourage us to relate to the relationship between computer science and technology in an unthreatening way. We should not conceal the fact that computers and technology are essential for enabling the evolving synergy between computer science and natural information processes. Moreover, presenting computer science as a language of technology illuminates essential linkages and cross-fertilization between both realms.

3. Educating the citizens of tomorrow

Computer science has increasingly become a core knowledge requirement for all educated citizens. Like the more traditional sciences, it provides an essential understanding of the world around us. Computers are part of almost every aspect of our lives and it is vital that we understand their capabilities and their limitations. It is our responsibility as computer science educators to assist all students in bridging the gap between using and understanding computers [8]. Computer science education should be planned for and adapted accordingly to various populations: (a) all students [14], (b) prospective students who wish to obtain expertise in the field [4,8], and (c) scientists of tomorrow [16].

Switching the focus – The description of computer science as a scientific domain that is a language of technology can attract newcomers. However, the field should not be portrayed to prospective students only in terms of "what it is" without convincing them that it provides the knowledge and skills foundation essential for contemporary technological advances [14] and may enhance understanding other subjects as well [7,17]. Computer science should be introduced as a high-level scientific language for problem solving, knowledge representation, and formalization of processes, as well as a language for understanding technology and performing technology-related processes.

4. References

1. Adler, A. Mathematics and creativity, in (T. Ferris, ed.), *The world treasury of physics, astronomy and mathematics*, Little, Brown and Co., 1991.
2. Chen, D. & Stroup, W. (1993) General System Theory: Toward a conceptual framework for science and technology education for all. *Journal of Science Education and Technology,* 2(3), 447-459.
3. Denning, P.J., Comer, D.E., Gries, D., Mulder, M.C., Tucker, A., Turner, A.J., and Young, P.R. (1989). Computing as a discipline, *Communication of the ACM*, 32(1), 9-23.
4. Denning, P. J. (2004). Great principles in computing curricula. *Proceedings of SIGCSE'04*, Norfolk, Virginia, USA, 336-341.
5. Denning, P.J. (2005). Is computer science science? *Communication of the ACM*, 48(4), 27-31.
6. Denning, P.J. (2007). Computing is a natural science. *Communication of the ACM*, 50(7), 13-18.
7. Guzdial, M. and Soloway, E. (2003). Computer science is more important than calculus: The challenge of living up to our potential, *inroads – SIGCSE Bulletin*, 35(2), 5-8.
8. Haberman, B. (2006). Teaching computing in secondary schools in a dynamic world: Challenges and directions, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 4226, 94-103.
9. Johnson, J.R. (1989). Technology: Report of the Project 2061, Phase I Technology Panel. Washington, DC.: American Association for the Advancement of Science.
10. Klawe, M. (2005). Changing the image of computer science- A North American perspective in conversation with Europe, Keynote, *Proceedings of ITiCSE'05*, June 27–29, 2005, Monte de Caparica, Portugal, 3.
11. Koffman, E., Ellis, H., Kelemen, C. White, C., and Wolfman, S. (2007). New paradigms for introductory computing courses. Proceedings of SIGCSE'07, March 1-10, 2007, Covington, Kentucky, USA, 67-68.
12. McGettrick, A., Cassel, L., Guzdial, M., and Roberts, E. (2007). The current crisis in computing: What are the real issues? Proceedings of SIGCSE'07, March 1-10, 2007, Covington, Kentucky, USA, 329-330.
13. Naughton, J. What is 'Technology'? in: (Banks, F. ed.) Teaching Technology, 1994, Routledge; London and New-York, in association with the Open University.
14. Stephenson, C., Gal-Ezer, J., Haberman, B., and Verno, A. (2006). The new educational imperative: Improving high school computer science education. Final report of the CSTA Curriculum Improvement Task Force February 2005, Computer Science Teachers Association, Association for Computing Machinery, http://csta.acm.org/Publications/White_Paper07_06.pdf [Accessed April 2007]
15. Smith, D. (1998). Computerizing computer science. *Communication of the ACM*, 41(9), 21-23.
16. Towards Science 2020. (2006). Microsoft Research. http://research.microsoft.com/towards2020science/downloads/T2020S_ReportA4.pdf [Accessed 6 May 2007]
17. Wing, J.M. (2006). Computational thinking. *Communication of the ACM*, 49(3), 33-35.