The Education Column

BY

Juraj Hromkovič

Department of Computer Science
ETH Zürich
Universitätstrasse 6, 8092 Zürich, Switzerland
juraj.hromkovic@inf.ethz.ch

How to convince teachers to teach computer science even if informatics was never a part of their own studies

Juraj Hromkovič, Regula Lacher
Department of Computer Science, ETH Zurich
ETH Zentrum, Universitätstrasse 6
8092 Zurich, Switzerland

Abstract

Computer science is becoming a common, mandatory subject in curricula of educational systems in ever more countries. The implementation is everything, but simple. Computer science is the only subject that teachers of primary schools have to teach, but never studied themselves. The widely spread, but wrong idea that computer science is about using a computer or about working with social media makes this process still more complicated, with the risk that the next generation of teachers also does not get proper training. In this article, we show how to explain the goals of computer science education to the teachers in such a way that they understand the contributions of computer science to the understanding of the world and to the growth of intellectual abilities of their pupils, and that they focus on teaching fundamental, and therefore stable concepts of informatics instead of operating instructions for short-term applications. The following text is an explanation and an offer to teachers who are experienced in teaching, but do not have any idea what computer science is about, except that it has "something to do with computers."

1 What is Computer Science?

Computer science is about automation of procedures. Therefore, despite the fact that computer science is considered to be a young scientific discipline, the way of thinking in computer science was a part of human culture since ever. Not only was the first known writing developed about 5 000 years ago by the Sumerians in order to solve a typical computer science task: find a method for storing and efficient processing of data about properties and duties (taxes) of about 1 000 000

inhabitants of Mesopotamia; but also because of the two basic concepts that enabled the growth of human societies which were – and still are today – the ability to create knowledge and the application of existing knowledge to achieve different goals. To become an expert in performing a (previously developed) procedure (algorithm, we could say today) was much easier than to learn to develop such procedures and to discover new knowledge. One wonderful example from ancient Greece was the Theorem of Pythagoras that became a procedure applied in the construction of houses. Since $3^2 + 4^2 = 5^2$, one knows due to Pythagoras that the triangle with sides of length 3, 4 and 5 units (it does not matter which units are used) contains a right angle. Therefore, to create a right angle it is sufficient to take 3 ropes of length 3, 4 and 5 units to build the corresponding triangle. There is a huge gap between the qualification of creating this triangle and the intellectual potential to discover the Theorem of Pythagoras, not to mention the capability to understand why it is true. In this way, many technologies became available for big parts of society in spite of the fact that only a few people were able to develop them.

Computer science became a distinguished discipline when the following two prerequisites were satisfied:

1. Many procedures (algorithms) were mastered and unambiguously described on a detailed level such that no intellectual ability of improvisation (expert knowledge) was required to perform them. Executing such procedures became a routine work.

2. Technologies were developed that enabled to delegate the execution of precisely described procedures to machines.

Because of that, two main components of computer science today are the following ones:

1. To work together with specialists in all other areas of science and human activity in order to develop efficient algorithms for automation of further human activities.

2. To contribute to the engineering of developing hardware and software technologies in order to be able to automate more and more complex tasks and to increase the performance of our computing technologies. A part of this job is the development of programming languages that enables humans to communicate with machines, to control them, and to instruct them to execute different activities.

2 Goals of Teaching Computer Science

To recognize truly valuable goals for teaching computer science, one has to follow the genesis of computer science thinking and its contribution to the development of human society. One is not allowed to focus on the latest hardware and software products and their applications. On one hand, these will change soon, and on the other hand, it is not a good idea for teachers to compete with their pupils in the knowledge about the most recent applications. The primary task of schooling is to support the development of the intellectual potential of young people. Simply learning to use different products of software industry does not really train our brain for creative work and, together with the corresponding multitasking, can even decrease our ability to concentrate and lead to intellectual underdevelopment.

Instead, our global goals are as follows:

I. To strengthen the fundamental competences in mathematics and languages.

II. To understand the technological world created by humans and to be able to contribute to its development.

III. To transform the constructive and creative thinking of engineering into school education.

I To contribute to fundamental competences in language and mathematics

The contributions to teaching mathematics are more transparent because the interface between mathematics and computer science is really big and multidimensional. Already data representation and the description of computing tasks require the exact language of mathematics.

The focus is primarily in contributing to the ability to solve problems. In contrast to classical education in mathematics, one does not aim at learning a given method, optimized for many years, for solving a problem as a final product of scientific work. In algorithmics and programming, one is always aware of the existence of many ways for solving a given problem. The goal in education must be to discover some of them and then to verify their correct functionality. In computer science terminology, a problem consists of a huge number of problem instances. For instance, the problem of solving quadratic equations consists of infinitely many problem instances – concrete quadratic equations $ax^2 + bx + c$ for arbitrary numbers a, b and c. The way of teaching computer science means that one has to collect experiences by solving concrete problem instances in order to discover a robust strategy working for all instances of the problem. The goal is

not necessarily to finish with the best-known strategy as a product of an optimization based on many years of scientific work. Mainly we focus on the process of discovering a functioning strategy. For sure, one is allowed to define criteria for measuring the quality of algorithms developed and to try to optimize them with respect to the chosen criteria.

Hence, the focus is not on learning some final products of science, but on learning the processes of their discovery, the processes of knowledge generation. One starts with a transparent motivation and learns to verify and correct own products.

If one understands mathematics as a language for the exact description of objects and different aspects of reality on one hand, and as a language of verifiable argumentation on the other hand, and in this way as a powerful research instrument, then teaching computer science has to contribute essentially to both of these fundamental competences and to developing research instruments that must be the kernel of any good education in science.

At first glance, contributing to education in languages by teaching computer science may surprise and because of that we have to explain it in detail. We distinguish between three dimensions that cannot be considered as completely independent. The first dimension is related to the development of writing. The second one to the development of language by introducing new words and the last one to using language for well understandable communication that avoids misunderstandings.

Let us first look on the development of writing because the whole computer science is based on representing all data (information) as sequences of symbols. One first chooses an alphabet as a set of suitable symbols, and then builds words as sequences of symbols and assigns an exact meaning to each word. The creative aspect of this process in computer science is that one is asked to develop different writings with respect to the purposes one aims at. There exists a variety of secret writings, special writings for reading texts that are able to automatically correct misprints and insert missing symbols, writings for representation of numbers, writings for coding data in the shortest possible way, or writings for a data representation that enables an efficient search in a huge amount of data. Teaching all this, one can learn to understand the process of developing writings and, moreover, due to creating own writings, pupils can imagine the processes of the natural languages creation to some extent.

Pupils usually view a language as a final product of human society that they are asked to learn. A language has a syntax (formal grammar) on one hand and a semantics on the other hand. Teaching programming means also teaching a language, called programming language, that computers understand and that can be used to control them. The basic building blocks of a programming language are also words that have an exact meaning. A program is a text that explains

to the computer some activity the computer has to perform. A good teaching of programming is not allowed to teach a programming language as a final software product. One starts with a few words and discovers the need for more words in order to be able to express whatever one wants. One has to teach how to teach the computer to understand new words and then to use them in the communication with the machine. One can see here the analogy to the development of natural languages. The crucial point is that all pupils are enabled to create new words and develop the programming language in their individual way. In this way, one trains to choose and introduce new words in such a way that one communicates in a transparent and succinct way.

The third dimension trains the communication in natural languages. One learns to describe procedures in an exact way avoiding any misinterpretations. The ability to express everything as a exact as possible is also needed for describing computing tasks and the way in which data are represented. This covers abstract descriptions of objects, situations, relationships, etc. by symbols and graphs.

II To understand the technological world and to be able to control it and contribute to its development

Humans discover knowledge not only in order to understand the world around them, but also to reach some goals. The results of this effort are technologies (such as fire, boats, the wheel, house construction, writing, printing, machines, computers, the internet, etc.) that always essentially increased the speed of the development of the human society. All school subjects have to contribute to the understanding of the development of technologies. The special task of computer science education is related to the construction and to the control of the technical world. At the very end the goal is to educate makers and not only consumers of digital technology.

Teaching computer science avoids seeing the technical world as a world of miracles that happen when one pushes the right button. Computer science education explains the functionality of technical systems, the ways one can build them and control them. Additionally, the pupils learn to do it by their own. Finally, they are able to automatize more and more complex activities.

III To introduce the way of thinking in engineering to schools

Human society achieved progress due to the combination of knowledge generation and creative, constructive activities. All new technologies were developed by an intelligent application of knowledge derived beforehand and due to experience one acquires by trying to create something in a constructive way. The main resources

of a potential success were, in different proportions from case to case, on the one hand, knowledge as something one fully understands and can precisely specify, and on the other hand, "know-how" of an expert as something one cannot explain in detail. One can understand very well why, in the time of technical revolution, the educational systems preferred to teach well specified and understood facts and methods and not the "fuzzy know-how" of technical disciplines. For the former, it was easy to define the goals and to measure the progress achieved. For the latter, nobody was able to specify the goals exactly enough for a school subject.

Today we have a completely different situation. Most competences related to methods and applications can be automatized and so one does not need humans to execute them. Society does not need experts able to execute maybe complex, but automatable activities. One needs experts that can do a creative work. They can be trained for this purpose and their expertise can grow with the number of projects in their training. It does not matter that measuring their progress in becoming experts is not easy.

On the other hand, computer science was able to formalize a lot of handling based on experience in technical disciplines, and introduced strategies that can be well explained. This enables to introduce the related topics to pupils. The main goals are as follows:

- To discover algorithms (methods) for solving problems by trying to solve several concrete problem instances.
- To implement the discovered methods in software (programming) or hardware.
- To test the functionality of their own products and to measure its properties such as efficiency and transparency.
- To optimize own products or to extend them to additional functionalities.

The education in the digital age cannot miss teaching these fundamental human competences of creative work starting with motivation and problem formulation and finishing with useful products. The students have to view never ending testing and iterative improvements as a natural and common process. They have to master the modular design. One starts with small programs called modules whose functionality is simple and can be easily verified. One uses these modules to build more complex programs, etc. It surprises even experienced educators that small pupils are able to build really complex systems in this way.