Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects*

Giovanni Serafini

Department of Computer Science, ETH Zurich, Switzerland
giovanni.serafini@inf.ethz.ch

Abstract. The key contribution of computer science to general and school education relies on the concept of Computational Thinking. Teaching programming in Logo at the primary school is an appropriate didactic approach towards Computational Thinking, it permits to embed Computational Thinking into a spiral curriculum at a very early stage and should enable specific transfer to related school subjects. The paper describes our concrete experiences in teaching programming in Logo at Swiss primary schools, reflects on didactic visions and consider prospects for long-term empirical research.

1 Introduction

Computer science is nowadays omnipresent in real life. University programs in computer science were already introduced in the late sixties of the last century and are actually well established. It is to be considered a kind of paradox that despite of its everyday relevance and its importance for university education and research, efforts aiming at introducing a dedicated computer science school subject evoke controversial discussions among school communities as well as among the scientific community [10]. A missing general consent about goals, teaching topics, and even about the scope of computer science and the delimitations to related disciplines characterizes the debate.

Jeannette Wing brings order into the discussion declaring Computational Thinking as the key contribution of computer science to general and thus to school education [11]. According to Wing's vision, we believe that Computational Thinking is an attitude as well as an extensive framework of concepts, abilities, and skills young people should learn in school. We are convinced that learning programming on an adequate level of abstraction is a very effective didactic approach to Computational Thinking, independent of the age of the pupils.

In this paper, we focus on our school projects aiming to introduce primary school pupils to Computational Thinking by teaching them how to program. The pupils usually attend grade 3 to 7 and are roughly between 8 and 13 years

* This work was partially supported by the Hasler Foundation.

I. Kalaš and R.T. Mittermeir (Eds.): ISSEP 2011, LNCS 7013, pp. 143–154, 2011.

old. During the last 6 years we were allowed to teach several hundred children programming in Logo, during on-site school projects as well as in dedicated events directly at our university.

The paper is organized as follows: Section 2 addresses the nature as well as the scientific fundamentals of Computational Thinking. Section 3 focusses on the adopted didactic approach while teaching programming at the primary school: we discuss the didactic and technical requirements for a programming language for very young pupils, present the vision and the didactic approach we follow, and further give a concise overview of the teaching materials we developed. In the following section, the structure and the timeline of a typical school project are shortly addressed. In Section 5, we reflect on general experiences and learning achievements during the school projects. Section 6 highlights research prospects dealing with Computational Thinking and with starting programming courses at primary schools, and finally outline prospects for empirical research.

2 Addressing the Nature of Computational Thinking

2.1 Computational Thinking Is Unique to Computer Science

In a seminal contribution, Jeannette Wing highlights Computational Thinking as a new thinking paradigm, which enables innovative ways for addressing problems in everyday life as well as for approaching research subjects in apparently completely unrelated scientific fields. Wing assesses that everyone, not only computer scientists, would be eager to learn and to use it [11].

Juraj Hromkovic explains in detail, that computer science has aspects of mathematics, natural sciences, technics and even of philosophy, but is not exactly included in one of these. He highlights that computer science is an independent scientific discipline which formally studies ways to automatically (algorithmically) solve problems, and assesses that the concept of algorithm should be considered as the first axiom of computer science. Since algorithms are the core of computer science, Computational Thinking is the key contribution of computer science to general education [3,5].

According to these two very similar definitions, algorithms are unique to computer science, and computational (or algorithmic) thinking can only be learned in a dedicated computer science school subject.

2.2 Children Should Learn Computational Thinking for Life

Computational thinking is a scalable concept whose extent can be adapted, depending on the abstraction skills and on the prior knowledge of the involved actors. We think that Computational Thinking can be introduced and taught on an appropriate and adequate level of abstraction at all school stages, and that it is well suited to be taught in a cognitive actively form of learning, within a spiral curriculum. We are firmly convinced that learning how to program represents a scientific sound, but very effective didactic approach to Computational Thinking, independent of the age of the pupils.

We believe that Computational Thinking is nowadays essential and that children should learn it for life. We therefore promote the idea that a computer science subject relying on Computational Thinking should be mandatory at every school stage, including the primary school. We do not pursue the goal to make a computer scientist or a software engineer out of each child, but we assume that introducing programming at a very early school stage might help increasing the interest in computer science and MINT subjects[1]. We hope that therefore, on a long-term perspective, more school-graduates would consider to enrol for a university program or to start a career in a MINT field.

3 Didactic Concept and Teaching Approach

3.1 The Quest for the Programming Language

We believe that Logo [9] still is one of the most adequate programming language for beginners, particularly for classes at primary school. Logo is a mini-language and, in contrast to general-purpose programming languages, it was explicitly developed for teaching programming [4]. In order to write and run Logo programs, beginners do not need to learn a consistent language subset and do not have to deal in any way with the intrinsic complexity and the particular features of a larger, general-purpose programming language [1]. Logo permits the user to focus on a very reduced set of instructions with an adequate, clear syntax. In contrast to sophisticated programming environments for other, well established mini-languages, simple Logo editors do not rely on a click-and-drag approach and allow the pupils to type the instructions by themselves. This permits them to care for correct syntax and dramatically reduces the cognitive overload caused by an unmanageable list of instructions or by an overcharged graphical user interface.

Moreover, the open-source programming environment XLogo [7] satisfies all our most relevant expectations. It is free of charge, it runs on multiple platforms, it does not need an internet connection at run-time and runs quite well on slow computers. Schools are therefore able to simply deploy it, even in older computer rooms, teachers as well as pupils and their families are allowed to use it without further constraints on their private computers.

3.2 Didactic Approach and Teaching Materials

The German textbook *Einführung in die Programmierung mit Logo* [4,2] is a detailed introduction to programming in Logo permitting to intensively teach classes of different school stages on a regular basis. The contents of the first part of the textbook are well suited for teaching programming at primary schools.

School projects aiming to introduce children to programming usually have very restricting time limitations. We therefore decided to develop compact, ad-hoc-teaching materials for primary schools adhering to these constraints which

[1] MINT is an acronym for mathematics, informatics, natural sciences, and technics.

still rely on the didactic guidelines from the mentioned textbook. The teaching materials are meant to be the ideal support for our school projects and have further made been available online, free of charge, for interested schools and institutions [6].

The teaching materials are organized in six different lessons and do not require prior knowledge in programming.

3.3 Lessons 1 to 4

The first lesson introduces the concept of a **program as a sequence of simple computer instructions**. The pupils learn the exact meaning of the four basic instructions *fd*, *bk*, *rt*, and *lt* which permit to move and to rotate the turtle, as well as the instruction *cs* used to clear the screen. They learn that the computer does only understand clear, unambiguously formulated instructions, and they further get introduced to the simple, but very effective XLogo-programming environment.

Lesson 2 focusses on the concept of a basic loop with a constant, a priori known number of iterations. The children are lead to discover the need for a way to automatically repeat a specific code segment, and learn the exact, but not yet very intuitive syntax of the *repeat* instruction. They start therefore writing shorter, clever programs which reuse code they already wrote and directly influence the sequential execution flow. At this stage, children do not need to be introduced to the concept of a variable.

Lesson 3 introduces the approach of modular design of programs, which is not unique to computer science and represents an ideal bridge to other technical disciplines. First of all, the pupils learn that programs mostly need a name and that a named program can be nested in a larger program. In a small project, pupils have to draw a town consisting of identical streets of exactly the same house type. The children recognize that modularity is a very powerful concept, permitting to conceive and produce programs solving complex assignments. They understand that modular design simply reuses existing programs in a very elegant way. The assignments need the two additional instructions *penup* and *pendown*.

Lesson 1 to 3 allow us to highlight some characteristics of the chosen didactic approach and of the teaching materials:

- There is no systematic direct instruction of the children by a teacher. The pupils are expected to work autonomously, relying on the teaching materials and on the interactions with the tutors.
- The theory blocks and the texts are very short, the language is simple, but the terminology is precise.
- Pupils are able to write and execute their first program within minutes. They only need to know a few instructions.
- The exercises of the first two lessons are simple and permit the pupils to familiarize themselves with the programming language and the basic concepts we introduced.
- The exercises focus on small algorithmic problems: the assignments are concise and the expected solutions are compact. Pupils are even able to verify

their solutions by themselves, on the screen, without a formal feedback of a teacher. If not, they can autonomously start debugging the code.
– A didactic sound approach allows to rapidly introduce basic concepts such as program, loop and modular design.

The fourth lesson deals with drawing regular polygons and circles. The new instruction *setpencolor* permits to change the color of the pencil. In this lesson, no new programming concepts are introduced. The pupils are mainly expected to gain deeper routine in programming by solving assignments for which they generally have prior knowledge from geometry and mathematics.

3.4 Lessons 5, 6 and the Concept of a Constant Parameter

Lesson 5 introduces a substantially simplified concept of a variable in form of a constant parameter. Relying on a placeholder abstraction and lead by the teaching materials, pupils learn that programs are usually able to solve a class of problems whose instances only differ in the value of one or more specific parameters. The initial example focusses on drawing a square. The exercises permit to vary the figure the pupils have to draw and to change the number of parameters. This approach allows to efficiently mask the hardware layer and therefore to reduce the intrinsic complexity of the concept of a variable to an appropriate level for primary school kids.

In Lesson 6, the pupils learn to combine parameters with modular design. They write challenging programs expected to pass parameter values to subprograms, integrating all the basic concepts and the instructions of the lessons 1 to 5.

The didactic approach behind lessons 5 and 6 is similar to the approach we chose for the first four lessons. Even if contents and exercises are now more complex, the pupils are still writing small programs based on very few instructions, in a simple and effective programming environment. The pupils can still learn, program, test, and debug in an autonomous way, while tutors assist them.

4 Programming Projects at Primary Schools

4.1 School Projects

Practical organisational constraints impose a very similar but flexible plan for each new school project. A typical school project currently consists of 12 to 16 units of 45 minutes each, which are usually taught in 3 to 4 half-day blocks. Classes are split into parallel groups of up to 12 pupils. Two tutors and usually one class teacher supervise each group. Each child uses an off-the-shelf-netbook, learns and programs autonomously and is therefore implicitly able to adapt his pace. Each school project ends with a programming contest which give us a more concrete impression of the learning achievements of the pupils. A closing act give us a feeling of the attitude of kids, teachers, school board and political authorities with respect to programming classes.

4.2 Class Teachers and University Tutors

We explicitly involve the teachers at the host-schools and expect them to actively support their classes during the school project, but we have no particular requirements with respect to their prior knowledge in programming. According to the mission of the primary school, classes are usually taught on a one-teacher basis and teachers are therefore required to integrate wide-ranging school subject competence with deep pedagogical and didactic skills. Their education in computer science mainly focusses on mastering computer applications which support the preparation and the documentation of their lessons. Consequently, the first project stage consists of training the teachers. This usually happens on-site, during a half-day workshop, relying on exactly the same didactic approach and the same teaching materials we later use with their pupils.

The school projects still involve tutors from our university. The composition of the team varies and comprises faculty members, lecturers, postdocs and PhD candidates, as well as several undergraduate computer science students. Female and male tutors are usually equally represented.

5 General Experiences

In this section, we present selected major observations from our school projects. These observations are common to most of the school projects and enable us to summarize our experiences in a general, meaningful way. Furthermore, we consider the results of a survey among pupils and teachers taking part in recent projects, present the assignments of a programming contest, and comment the results. The reflections generally focus on didactic issues to be additionally addressed and eventually empirically investigated.

5.1 Major Observations

- During each project, the pupils need up to 30 minutes at the beginning of day 1 in order to get used to learning autonomously.
- Pupils get rapidly used to the Logo syntax. Typing the code by themselves is not a particular issue.
- The pupils are really dedicated. They visibly like programming and are still proud to show their progress and the solutions to the assignments. They usually have to be forced to leave the classroom during the mandatory breaks.
- The kids always try to improve their solutions. The are not satisfied with a weak solution.
- Girls are as engaged and achieve the same goals as boys.
- Starting with lesson 4, the cognitive complexity of the taught contents increases. Pace differences among the pupils increase, too. Since the pupils are working autonomously, tutors do not have to assist the whole group but are able to actively support kids having particular problems or looking for being challenged.

- Teachers are really motivated, even if they do not have any kind of prior knowledge. They enjoy to learn programming, like to have well-prepared teaching materials they can later use and like the exchange of insights about programming and didactic issues with university staff and tutors.
- The teachers are not necessary faster in learning than the kids.

5.2 School Project in Attinghausen

A recent project took place a the primary school of Attinghausen, in the Swiss Canton of Uri, near to the Saint-Gotthard Massif. 16 pupils of grade 5 and 23 pupils of grade 6 were taught in 4 parallel groups during 3 half-day blocks, in the morning of March 14th, 16th, and 18th 2011. Three weeks before, 4 teachers, the school director as well as a local politician were introduced to Logo within a three hours course. Each group of pupils was supervised by two tutors as well as by one class teacher. Every group acted independently from the others. One computer for each of the pupils was available. The last 90 minutes of the school project were reserved for a programming contest and for the common closing act.

The school board initiated a survey, mainly aiming to assess the impression among all the pupils and two of the teachers. Table 1 shows selected items from the survey. Each item was graded on a scale from 1 (I totally disagree) to 6 (I totally agree). The pupils and the teachers really enjoyed the project. Their feedbacks are very consistent. One teacher pointed out, that he would like to install XLogo at home, but he would need a short break in his time intensive job. Furthermore, we noted that not all the kids knew where they can download XLogo.

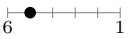
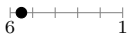
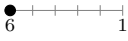
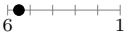
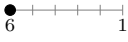
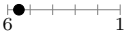
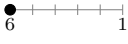
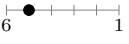
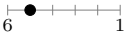
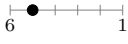
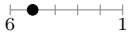
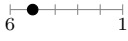
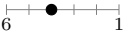
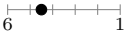
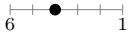
5.3 School Projects in Domat/Ems and in Saas im Prättigau

The Higher School of Pedagogics of the Swiss Canton of Graubünden is very interested in supporting the introduction of computer science at the local primary schools.

During two large pilot projects in 2010 and 2011, in Domat/Ems, our team was able to teach up to 120 kids of grades 5 and 6 during three half-day blocks, as well as to train their teachers. A summary of the 2010 project is available online [8]. The statements of the pupils and of the teachers are consistent with those of Attinghausen.

On May 12th, 19th, 26th and June 9th 2011 we taught a grade 5 class comprising 17 pupils at the primary school of Saas im Prättigau. The usual half-day blocks were split into two 90 minutes units just before and just after lunch. The large and comfortable classroom allowed to teach all kids together, with the supervision of two or three tutors and of the class teacher. One computer for each of the pupils was available. As usual, the last 90 minutes of the school project were reserved for a programming contest and for the common closing act.

Table 1. Selected items from the Attinghausen's survey

	grade 5	grade 6	teachers
I liked the project. I recommend it to other schools.			
The teaching materials are well-arranged and are comprehensible.			
I found it proper and important, that each pupil was allowed to individually use one computer.			
My perception of the coaching by ETH tutors and class teachers was positive. Tutors helped me by questions and supported me with hints.			
Working in four small groups, instead of the whole class, was helpful for this project.			
The closing act, with programming contest and dia-show was good and motivating.			
The distribution of the courses over 3 mornings was optimal.			
I like to occasionally program with XLogo and would enjoy to take part to a continuation of the school project.			
I plan to installed XLogo at home (or I already did it), so that I can continue to program.			

5.4 Programming Contests

The programming contests generally consist of six to nine tasks which are very similar to the exercises the pupils had to solve during the course and they are similar over all the projects. Table 2 shows the tasks of the programming contest we organized in Saas im Prättigau. The contests are neither a school examination nor an empirical test to assess the learning achievements of the pupils. They should help increasing the motivation of the children and allow to celebrate the end of the project.

We do not prepare an official ranking of the constants, but usually gift the participants solving all the tasks (or most of them) with a Logo textbook [4]. The pupils are free to choose the tasks they like to solve first and are allowed to submit their solution to a tutor several times, until they have the correct one. Therefore, we only reward correct solutions with one point. Only in very particular situations (e.g. when the time is running out and a child submits a good but not perfect solution) we distribute a fraction of a point.

Figure 1 summarizes the results for each of the tasks of the contest of Saas im Prättigau: 15 of the originally 17 pupils attended the contest. The pupils were expected to solve 7 exercises resp. 9 different tasks. The class reached an average of 6.4 out of 9 points. 60% of the children solved at least 6 tasks, 47% of them solved 7 tasks and 3 pupils were able to solve all the nine tasks. These remarkably good results confirmed the excellent feeling we had during the lectures.

The Attinghausen project consisted of only three half-day blocks. The programming contests was therefore easier and shorter. Only two tasks required to be solved using parameters. The two overall best participants were two grade 6 girls. They were able to solve all the six assignments very rapidly, within 31 resp. 33 minutes.

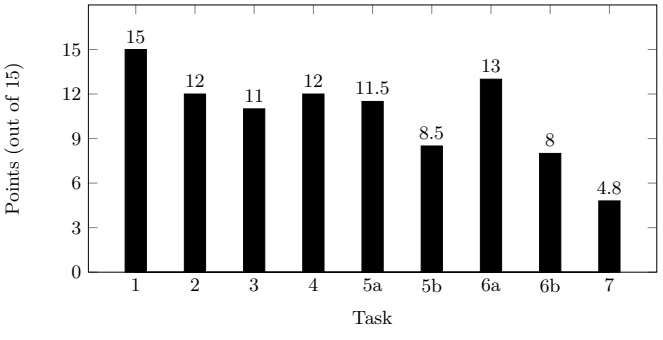


Fig. 1. Saas im Prättigau: Points per Task

6 Research Prospects

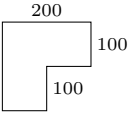
The very exiting and promising teaching experiences we made in the schools confirmed us in our conviction that extended, long-term empirical research on Computational Thinking should not be postponed. In this section, we first consider the general idea of assessing Computational Thinking, we focus on exemplary research topics dealing with the concept of a variable, and finally shortly point out our next steps.

6.1 Assessing the Influence of Computational Thinking

It can be reasonably assumed that the interdisciplinary nature of computer science may encourage a specific transfer of Computational Thinking to other scientific fields. Long-term empirical research projects are needed in order to formally investigate the impact and the probable benefits of increasing Computational Thinking skills on closely related school areas such as mathematics, natural sciences and technics. We believe that starting to program at a very early school stage plays a central role in learning Computational Thinking, and suggest to start programming classes, within a spiral curriculum, at primary schools.

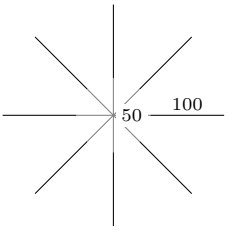
Table 2. Programming Contest of Saas im Prättigau

1. Let a program draw the following picture.

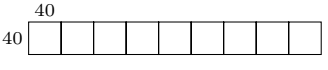


2. Draw the following picture.
Use `repeat`.

3. Write a program for this figure.
Use `repeat`.
The gray lines must not be drawn

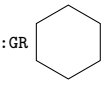


5. Fill in the necessary parts in the given programs in order to draw the picture.
a) `repeat 9 [repeat 7 [fd 40 rt 90] ...]`
b) `fd 40 rt 90 fd 360 rt 90 fd 40 rt 90 fd 40 repeat 8 [rt 90 fd 40 ...]`

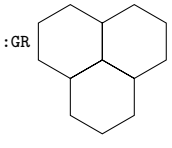


4. The program
`HALFCIRCLERIGHT`
`HALFCIRCLELEFT`
`HALFCIRCLERIGHT`
should draw the following picture. Write the subprograms `HALFCIRCLERIGHT` and `HALFCIRCLELEFT` which allow to achieve this goal.
You can freely choose the dimension of the semi-circles.

6. a) Write a program `HEXAGON :GR` to draw a regular hexagon with side length `:GR`.



b) Use `HEXAGON :GR` as a sub-program of the program `PATTERN :GR` to draw the following picture:



7. Write a program which draws the following picture. Users should be free to choose the side length of the squares and the number of black squares. There are as many gray squares as black squares.

In hypothetical experimental settings, a group of pupils receives an additional, excellent education in computer science, while the other groups independently get an additional, excellent education in closely related school subjects. A control group has no additional education at all. This or similar experiments, even experiments with additional education in not-MINT-topics, and the comparison of the overall learning progress of the scholar groups over several years may help addressing the influence of Computational Thinking in the context of general education.

Variations of these experimental settings allow to assess the impact of Computational Thinking skills on specific groups of pupils, e.g., female pupils, very young pupils, pupils with an high IQ or pupils with a low IQ.

6.2 Research on the Concept of a Variable

The concept of a variable is usually the first crucial issue for programmer novices of each age. Its cognitive complexity can be reduced by splitting the learning process into two phases. The pupils first learn to deal with a constant parameter, relying on a simple placeholder abstraction. The primary school kids of our school projects usually master this abstraction level. In a second step, eventually at a higher school stage, the pupils learn that parameter values may vary over time, and that a variable parameter is intrinsically related to computer memory. A simple, but adequate abstraction allows to represent it as a register. How old should children be at least, in order to master this approach, in particular the step from a constant to a variable parameter? What kind of abstraction skills or prior knowledge do they need? Is the concept of a variable parameter generally too hard for primary school kids?

Mastering variables in computer science may open new, even unexpected didactic perspectives. One of the most fascinating and challenging research prospect refers to the only apparently similar role of variables in computer science and mathematics. Children starting to program early, in the primary school, do not have any school related prior knowledge on variables. They therefore develop this concept from scratch. Even relying on the register abstraction, a variable in computer science is more concrete than in mathematics. Children who first learn to program and later have to deal with the more abstract concept of variables in mathematics classes, may benefit from their programming skills.

6.3 Upcoming Research Activities

Gaining empirical evidence of learning progress requires a sound test design as well as a large population of pupils to be followed over a long period. The pupils have eventually to be followed over most of their mandatory school, and even longer. We are looking ahead for planning and soon starting long-term research projects with partner institutions and an interdisciplinary group of scientists and teachers. Highly interested primary schools are already available. Classes at higher school stages have to be found.

7 Conclusion

This paper considers Computational Thinking as the key contribution of computer science to general education. It presents our experiences in introducing primary school kids to Computational Thinking by teaching them how to program. In our projects, primary school kids are expected to learn autonomously, without a systematic direct instruction. The pupils solve small algorithmic problems by programming in Logo, relying on very few commands in a simple editor. They are therefore much less exposed to the undesirable effects of cognitive overload.

Even relying on an appropriate sound didactic concept, programming remains a very challenging cognitive activity. Modular design is one method the kids learn

in order to address this complexity. Observation during the courses, the results of the programming contests, and the feedback of pupils and teachers informally confirm that the kids achieve the goals we set, and point out the quality of the chosen didactic approach, independent from the gender of the pupils.

Computational thinking may permit specific transfer from computer science to the other MINT subjects, and may eventually open new didactic opportunities. Extensive research projects are needed to gain empirical evidence of the its importance.

References

1. Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., Miller, P.: Mini-languages: a way to learn programming principles. Education and Information Technologies 2, 65–83 (1998)
2. Freiermuth, K., Hromkovič, J., Steffen, B.: Creating and testing textbooks for secondary schools. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 216–228. Springer, Heidelberg (2008)
3. Hromkovič, J.: Contributing to General Education by Teaching Informatics. In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 25–37. Springer, Heidelberg (2006)
4. Hromkovič, J.: Einführung in die Programmierung mit Logo. Vieweg+Teubner (2010)
5. Hromkovič, J.: Informatik und allgemeine Bildung (May 2010), `http://www.educ.ethz.ch/unt/um/inf/all_inf/unt/um/inf/all_inf/`
6. Hromkovič, J., Keller, L., Serafini, G., Steffen, B.: Programmieren mit Logo, `http://abz.inf.ethz.ch/primarschule-unterrichtmaterialien`
7. Le Coq, L.: Xlogo. Website, `http://xlogo.tuxfamily.org/`
8. Matter, B.: Projekt programmieren in der primarschule. Website, `http://abz.inf.ethz.ch/media/archive1/programmierenfuerkinder/InfobroschAug2010-3.pdf`
9. Papert, S.: Mindstorms: Children, Computers and Powerful Ideas, 2nd edn. Basic Books, New York (1993)
10. Schnabel, R.B.: Educating computing's next generation. Commun. ACM 54, 5 (2011)
11. Wing, J.M.: Computational thinking. Commun. ACM 49(3), 33–35 (2006)