The Education Column

by

Juraj Hromkovič and Dennis Komm

ETH Zürich, Switzerland

`juraj.hromkovic@inf.ethz.ch` and `dennis.komm@inf.ethz.ch`

How Teaching Informatics Can Contribute to Improving Education in General

Juraj Hromkovič and Regula Lacher
Department of Computer Science, ETH Zurich
juraj.hromkovic@inf.ethz.ch, regula.lacher@inf.ethz.ch

Abstract

Current education focuses on teaching facts, models, methods, and skills–and applying them to solve different tasks. Due to computer science, most of these tasks can be automized nowadays. As a result, future education has to change its focus on supporting students in exploring their creativity and on training critical thinking. In this article, we first explain how one can support the development in this direction, and then show by some examples that well-designed computer science textbooks can take a pioneering role in this process.

1 Concept for Teaching in the 21st Century

The main goal of education is the holistic development of the students' personalities (taking into account their individual talents) with the aim of maximally supporting the development of their potential, thus ultimately contributing to the development of society. To learn means to build one's own new concepts in context (abstraction and modeling), to think critically, and to be creative. Schools are expected to prepare their graduates for future careers–and more broadly for the future world.

However, in the 21st century, no one can reliably predict what kind of competences will be asked in professions in 20 years. All we know is that everything that is understood to some degree will be automated. Learning to operate equipment, apply predefined methods, or train certain skills will have less and less educational value in the future. Digital technologies already make it possible to perform these activities more accurately, faster, and more reliably. The only advantage humans have over man-made technologies is critical thinking, creativity, and the associated ability to improvise and redesign. Therefore, schools in the 21st century must focus primarily on the emotional and intellectual development of students through activities that foster their creativity and critical thinking.

Almost all educational systems to a high extent remain stuck in the paradigm of educational requirements from the time of the industrial revolution of the 19th century. In that time, the labor market needed experts who were able to apply complex procedures (methods, algorithms) to solve various tasks. However, these tasks are now largely automated. This also corresponds to the content of today's textbooks, which were written with the aim of imparting knowledge and practicing certain skills (following operating instructions and executing algorithms). In this sense, this is only a higher form of memorization. It is not enough for the school to teach developed concepts, teaching must encourage and enable the students to walk on the paths of discoveries and innovations to rediscover basic concepts, notions, methods, and technologies.

The current state of the educational system cannot be changed in the short term. It is necessary to set in motion an evolutionary process that, in a longer sequence of steps, "gently" transforms the current educational model, focused on teaching prefabricated knowledge and certain skills, into an education that develops the potential of students in dimensions that will not be replaced by automation and technology. The motto for teaching and textbook design must be:

> Do not teach the products of science (facts, theorems, methods, models, technologies) and the skillful use of them, but the processes of their discovery and creative development.

This approach is in line with the current international trend of acquiring competence. Competence is not the ability to solve routine problems according to a learned method. Competence is the expertise with which we are able to act meaningfully and originally in new situations based on our knowledge and experience.

Graduates of such training are truly innovative, with an intrinsic motivation to discover, understand, and create. They do not believe any statement presented to them, but they question those statements independently, examining their genesis and rationale.

Criteria For the Creation and Evaluation of Textbooks

The classical, common criteria of technical correctness and comprehensibility remain the basic prerequisites for good teaching. However, these criteria are not sufficient to achieve the above goals. The key to initiate the process of innovation in teaching is to formulate criteria that will help teams of authors to develop textbooks with the above goals in mind. First and foremost, the quality of any textbook—and thus teaching—should be measured by the extent to which it can support the achievement of the following goals:

1. Decomposition of the learning content and of its discovery process into a sequence of small steps, so that students have a realistic chance to go through the individual steps independently and thus to rediscover and master knowledge largely on their own.

2. Detailed and easy-to-understand explanations that allow for a mostly independent study or a study with minimal assistance. Each student, according to individual needs, can work through the topic any number of times at any individual pace. The examples and solutions to all tasks include not only the results and a brief description of the approach, but also the thoughts that clarify why the solver proceeded in the manner presented. If there are several possible ways of solving the problem, alternative solutions should be presented.

3. A motivating introduction to each topic to cognitively activate the class.

4. Support each student individually. An easy-to-understand approach to the basic knowledge and to the activities that must be mastered by all, as well as challenging topics for gifted and motivated students. The textbook must allow the teacher to differentiate the objectives in accordance with the abilities of each student.

5. The students should learn to a high degree through creative activities. The goal is to train the students to analyze the properties, functionality, and applicability of the products of their own work–and, in this context, to motivate them to improve their products.

6. Guide the students in conducting experiments and in analyzing the results of the experiments in order to acquire new knowledge.

7. Motivate and strengthen the will to try to solve problems and to learn from failed attempts (mistakes) that did not lead to the set goal(s).

8. Carefully cultivate precise terminology and to practice precise wording in all forms of communication.

9. Increase the sustainability of the newly acquired knowledge by placing it in the context of already available knowledge.

10. Promote teamwork with intensive communication while solving tasks.

11. Offer instruments enabling students to measure their progress independently.

All of the above goals taken together should "guarantee" the sense of accomplishment that is the most effective teaching method. The final objective must be the increase of self-confidence of the students, and their willingness to solve problems and to create new products on their own.

Writing textbooks which achieve the above goals requires a high level of expertise from the writing team in several areas–more specifically:

- In-depth subject knowledge in a broader and deeper context including the genesis of the development of the scientific discipline under consideration;
- pedagogical experience from teaching in the specific age group;
- and knowledge of subject didactics of the scientific discipline under consideration.

When creating textbooks, we recommend the use of the following elements:

- *Elements of cognitive activation for the introduction to the subject*–e.g., puzzles, surprising observations, seemingly contradictory statements, or attractive applications;
- *examples with presentations of approaches* and reflections on them;
- *learning tasks* with the aim of discovering new relationships (context) or finding a way to solve a problem;
- *analysis of and learning from failures*;
- *design of experiments, execution thereof, and analysis of results*;
- *project tasks* to solve problems in teams and to create products with the desired properties;
- *classic exercises* to consolidate what has been learned;
- *historical and social remarks* explaining the genesis of the studied topic in a scientific and social context;
- *questions* to verify the correct use of technical terminology;
- *frequent summaries* of the newly acquired knowledge in the context of the already established knowledge;
- and *test items* to independently test the acquired subject knowledge.

Each textbook should be accompanied by a teacher's guide, which should also be accessible to the students' parents. This methodological material should ensure the following:

- Provide teachers the knowledge transfer related to the subject to be taught (in a broader and deeper context);
- show teachers in detail how the lessons can be implemented, explaining exactly the objective of each element of the textbook and how to verify the success of the corresponding learning process;
- explain how the textbook can be used to guide students individually in the learning process;
- provide a clear specification of objectives and instruments for measuring the learning progress;
- give detailed solutions to the exercises including didactic comments;
- provide advice and recommendations on how to respond to unexpected original approaches proposed by students.

2 Teaching Informatics As a Pattern For Teaching Other Subjects

Why can teaching informatics take a crucial role in transforming education in the direction proposed in the first section? Let us consider *constructionism* of Seymour Papert [20–22], where the main idea can be expressed by the genius sentence "Learning by getting things to work." Starting by "Learning as building knowledge structures" of Jean Piaget [23], and continuing by "Learning by doing" of Aristotle [1] and Comenius [5], students construct some products (programs, secret codes, number representations, data organization, hashing functions, algorithms, etc.). But finishing the construction of their products is not the end of their activity, but rather the beginning of their learning process. The immediate next step is to investigate the functionality and the properties of their own products, not only to correct them if required, but especially in order to get new ideas on how to improve them or how to extend their functionality. This brings students on a new starting line of their creative activity with the goal to construct something better than they did before. This is exactly how humans live, work, and learn since the dawn of mankind: an infinite loop of improvements and motivations, each step enabling a deeper understanding and increased expertise. This is the only way to properly build up competences.

In what follows, we discuss only two concepts related to teaching programming and data security. For a large number of examples of such teaching sequences we, refer to the textbooks [2, 3, 8, 9, 12, 13, 15, 19] for students and textbooks for teachers [4, 6, 7, 10, 11, 14, 16, 18], covering all ages from kindergarten to high school.

Programming

Why do we program? Because we want to explain some activity to a computer or to a robot in such a way that the technical system becomes able to execute this activity autonomously. In these terms, a computer program is a description of an activity in a programming language that is understandable for the machine. Developing a program is a very creative and constructive activity that fits our main goals because:

- One has to find a strategy of how to reach a given goal. Thus, problem solving is the key issue. Students can first train to correctly interpret the problem description, then to develop their own descriptions of the problem by using concepts such as tables, graphs, equations, etc., and finally build experience by trying to solve concrete problem instances. After developing a solving strategy, they can test its functionality by searching for problem instances for which their strategy does not work, or look for arguments as to why the strategy works properly for all instances of the problem. This is like testing a model by experiments.

- After having designed an algorithm (solving strategy), students have to describe the algorithm by a program in a programming language. A program as a product of the students' own work can be analyzed with the focus on its functionality. This is one of the reasons why tasks for programming novices deal with moving in the plane or drawing a geometric shape in order to enable to visually study the execution of the program. If the program does not work properly, the children have to try to fix a logical error in the program description or revise their strategy. If the program works properly, the students can think about extending its functionality or about formulating more complex goals and use the program developed as a module (building block) for developing a new program for the more complex task.

- Advanced students can analyze their programs with respect to their computational complexity, compare the efficiency of different programs, and then start searching for an algorithm or an implementation, respectively, that is more efficient than those developed up to now.

- Trying to reach the competences of a good programmer, it is not sufficient to develop programs for different purposes. Students must be trained to analyze programs of others, to correct them, and to modify them for some new purposes. All of this again means a lot of testing (experimental work) in order to understand the functionality of the program investigated.
- A program is a syntactically correct text in a programming language. A computer can interpret and execute a program only if there is no syntactical (grammatical) error in the program. A program is a precise and complete description of an activity, and the program as a text has only one unique representation. Thus, writing programs is good training in the consequent application of the syntactic rules as well as in semantically correctly expressing what one would like to describe.

We see that the goals 1, 2, 5, 6, 7, and 10 of good teaching listed in Section 1 are implicitly included in well-designed programming courses. In addition, all the remaining goals can be achieved as well when the details of teaching sequences are designed properly. We did consider the first goal by using the historical method, i.e., following the genesis of programming languages. Students start to learn programming by using a very small vocabulary. We teach them how to create new words and explain their meaning to the computer. In this way, students take part in the development of the programming language, improve its expressibility, and test this product of their own work by programming.

Cryptography

Cryptography as design and analysis of secret writings is about 4000 years old, and it is the kernel of data security. Following the first goal of good teaching we again propose to apply the historical method. The main goal is not to show some examples of secret writings or cryptosystems and to learn to use them, but to present students some concepts for designing and attacking secret writings in order to encourage them to develop their own, original secret writings and to try to break systems developed by others.

One can start to teach this topic very early (starting already in 3rd class) due to the fact that the ancient rule of secrecy was that the description of secret writings must be so simple that anyone can learn it by heart, i.e., the description does not need to be saved in written form on a medium. Following [3, 6, 7, 11, 12, 15, 16] one can see that the secret of the first secret writings based on transpositions can be described by simple pictures, and so the students can develop secret writing by describing them by simple pictures as well. The same is true for substitution-based secret writing. In this case students design new secret alphabets in a funny way and explore their fantasy.

The study of the properties (quality) of the designed cryptosystems is then done by attacking them. First, students can learn to break the cryptosystem Caesar by analyzing the letter frequencies in given cyphertexts, and later they can learn to break any monoalphabetic cryptosystem by the analysis of the distribution of letter frequencies. Trying to make their own cryptosystems secure against the letter frequency analysis, students can reinvent the cryptosystem Vigenère. For designing small steps in this direction, one can again follow the historical development, in which the first step was to increase the key (shift in the alphabet) of Ceasar by 1 after ciphering each single letter. Vigenère is nothing else than jumping in the *tabula recta* following a special pattern. Another simple step towards Vigenère is to use different Ceasar keys for letters at even and odd positions in a given plaintext. One can again let students design their own polyalphabetic cryptosystems before approaching Vigenère. Breaking Vigenère can also be done in a sequence of small steps [3] and can be the invitation for moving from context-free cryptosystems to context-sensitive (block-based) ones.

Here is again a lot of freedom to design plenty of own cryptosystems that cannot be broken by the analysis of Vigenère. We stop at this point, because the description how to make modern cryptography understandable for high-school students requires much more space than we have available here. For a concise introduction of public-key cryptography in high schools, we recommend the paper by Keller et al. [17]. But the key point here is that we can teach secret writings in such a way that the students can develop a large part of crucial concepts by themselves, or one can present the development of other ideas understandable in small portions. Another reason why cryptography is very attractive to a class is that on the beginning of each new special topic one can cognitively activate the class by funny, challenging puzzles.

3 Conclusion

We presented two key computer science topics to show that they are predestined to be mastered by reaching the set educational goals and offer patterns of how to teach for other disciplines. It is important to note that all parts of informatics can be taught this way. What are the main activities of computer scientists?

- Developing object representations and codes with required properties (short, secret, self-verifying, efficient to handle, etc.);
- designing efficient algorithms for different purposes (solving problems, translating codes and languages, etc.);

- and controlling technology by programming and designing IT systems that offer numerous services.

In all these cases we can get expertise exactly in the way proposed. One designs and develops a product that can be analyzed with respect to the properties, functionality, and suitability in different applications under different conditions. The historical method can be applied to make all the knowledge available in small steps based on activities of the learners.

Acknowledgements

We thank Peter Borovansky, Martin Curuc, Dennis Komm, Viliam Kratochvil, Daniel Masarovič, Maria Smerkova, Gabriel Thullen, and Michal Winczer for interesting discussions and for their comments and on previous draft versions of this paper.

References

[1] Aristotle. *A New Translation of the Nichomachean Ethics of Aristotle.* J. Vincent. Oxford, 1835

[2] Jarka Arnold, Cédric Donner, Urs Hauser, Matthias Hauswirth, Juraj Hromkovič, Tobias Kohn, David Maletinsky, and Nicole Roth. *Informatik, Programmieren und Robotik (in German).* Klett und Balmer. Baar, 2021.

[3] Michael Barot, Britta Dorn, Ghislain Fourny, Jens Gallenbacher, Juraj Hromkovič, and Regula Lacher. *Informatik, Data Science und Sicherheit (in German).* Klett und Balmer. Baar, 2021.

[4] Michelle Barnett, Juraj Hromkovič, Anna Laura John, Regula Lacher, Pascal Lütscher, and Jacqueline Staub. *Einfach Informatik 1 / 2, Spielerisch Programmieren mit Robotern (in German).* Klett und Balmer. Baar, 2021.

[5] John A. Comenius. *Didactica magna.* 1675.

[6] Urs Hauser, Juraj Hromkovič, Petra Klingenstein, Regula Lacher, Pascal Lütscher, and Jacqueline Staub. *Einfach Informatik 1 / 2, Rätsel und Spiele ohne Computer (in German).* Klett und Balmer. Baar, 2021.

[7] Heinz Hofer, Juraj Hromkovič, Regula Lacher, Pascal Lütscher, and Urs Wildeisen. *Einfach Informatik 3 / 4. Programmieren und Rätsel lösen, teachers guide (in German).* Klett und Balmer. Baar, 2021.

[8] Heinz Hofer, Juraj Hromkovič, Regula Lacher, Pascal Lütscher, and Urs Wildeisen. *Einfach Informatik 3 / 4. Programmieren und Rätsel lösen, textbook (in German).* Klett und Balmer. Baar, 2021.

[9] Juraj Hromkovič. *Einfach Informatik 5 / 6. Programmieren, textbook (in German).* Klett und Balmer. Baar, 2018.

[10] Juraj Hromkovič. *Einfach Informatik 5 / 6. Programmieren, teacher's guide (in German).* Klett und Balmer. Baar, 2019.

[11] Juraj Hromkovič and Regula Lacher. *Einfach Informatik 5 / 6. Lösungen finden, teacher's guide (in German).* Klett und Balmer. Baar, 2019.

[12] Juraj Hromkovič and Regula Lacher. *Einfach Informatik 5 / 6. Lösungen finden, textbook (in German).* Klett und Balmer. Baar, 2019.

[13] Juraj Hromkovič. *Einfach Informatik 7–9. Strategien entwickeln, textbook (in German).* Klett und Balmer. Baar, 2018.

[14] Juraj Hromkovič. *Einfach Informatik 7–9. Strategien entwickeln, teacher's guide (in German).* Klett und Balmer. Baar, 2018.

[15] Juraj Hromkovič. *Einfach Informatik 7–9. Daten darstellen, verschlüsseln, komprimieren, textbook (in German).* Klett und Balmer. Baar, 2018.

[16] Juraj Hromkovič. *Einfach Informatik 7–9. Daten darstellen, verschlüsseln, komprimieren, teacher's guide (in German).* Klett und Balmer. Baar, 2018.

[17] Lucia Keller, Dennis Komm, Giovanni Serafini, Andreas Sprock, and Björn Steffen. Teaching public-key cryptography in school In *Proc. of the 4th International Conference on Informatics in Secondary Schools: Evolution and Perspectives (ISSEP 2010)*, volume 5941 of Lecture Notes in Computer Science, Springer-Verlag 2010, pages 112–123.

[18] Tobias Kohn and Juraj Hromkovič. *Einfach Informatik 7–9. Programmieren, teacher's guide (in German).* Klett und Balmer. Baar, 2018.

[19] Tobias Kohn and Juraj Hromkovič. *Einfach Informatik 7–9. Programmieren, textbook (in German).* Klett und Balmer. Baar, 2018.

[20] Seymour Papert. *Mindstorms: Children, Computers and Powerful Ideas.* Basic Books, Inc. Publ. New York, 1980.

[21] Seymour Papert. *The Children's Machine: Rethinking School in the Age of the Computer.* Basic Books, Inc. Publ. New York, 1993.

[22] Seymour Papert and David Harel. *Constructionism.* Ablex publishing. New York, 1991.

[23] Jean Piaget and David Harel. *The Psychology of Intelligence.* Cambridge, Harward University Press, 1950.