# The Education Column

BY

**Juraj Hromkovič and Dennis Komm**

ETH Zürich, Switzerland

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

# Testing of an Interactive Online Learning Environment with Focus on Algorithmic Tasks

Dario Naepfer
Department of Computer Science, ETH Zurich, Switzerland
`dnaepfer@student.ethz.ch`

## Abstract

In this paper, we present a concept of computer science education, its implementation in an interactive online learning environment, and the extensive testing thereof. The goal is to support the development of algorithmic thinking by interactively solving small input instances of computing problems. The four approached competences are as follows:

1. To understand the abstract problem description and give proof of comprehension by classifying solution candidates into feasible and unfeasible solutions.
2. To find a solution for a given problem instance.
3. To find several different solutions for a given problem instance.
4. To apply criteria to evaluate and compare solutions, and to search for optimal solutions.

The contribution of this paper is twofold: First, we design the didactic approach and implement a learning environment in accordance with the aforementioned competences. Second, we test the environment in schools under various circumstances and present results of live testing, survey sessions including written feedback, as well as empirical test data derived from survey statistics. The empirical test covers user experience, intuitiveness of the learning environment, task difficulty, as well as satisfaction. This provides a genuine analysis of the application of the learning environment and allows drawing conclusions on the effectiveness of the applied didactic concept.

## 1 Introduction

Since the very beginning of humanity, Computer Science has been an important part of human culture due to its common roots with mathematics and written languages. This provides a good reason to consider informatics as a pivotal concept of education. Furthermore, it also qualifies the idea to include informatics as a fundamental pillar in the curricula of schools.

Multiple scientists contributed to the characterization of the discipline of informatics: Nygard [14] defined informatics as conceptual modelling and information systems. By Harel [8], informatics was depicted as a discipline covering computational, behavioural, and cognitive complexity, whereas Denning and Rosenblum [6] classified informatics as one of the four principal domains of science. Hromkovič and Lacher [12] further expanded the existing definitions as they added abstraction and symbolic representation, providing more efficiency. By describing the three roots of informatics, they presented a more holistic view of the discipline in the context of science than earlier descriptions, providing an overview of the potential of informatics education.

We define algorithmic thinking following Serafini [15]: the ability to systematically find solutions for problems with automated solution methods based on an iterative procedure. This makes it possible to extract, model, and represent information such that it can be written in terms of finite series of symbols chosen from a given set. Further, it characterizes the method, provides arguments supporting correctness, and analyses the amount of resources needed for execution. Also, it allows for a third party (e.g., a computer) to execute the solution method.

The learning environment developed here [13] is related to the concepts presented in *Computer Science Unplugged* [2,3], *"Abenteuer Informatik" (Computer Science Adventure)* [7], or *Algorithmic Adventures* [11]. Our environment is based on the concept of the textbook *"einfach Informatik 3/4" (Simply Computer Science)* [9, 10]. The authors present their as follows [5]:

> The starting point is merging constructionism and critical thinking. Constructionism with its "learning by doing" and "learning by getting things to work" enables designing a teaching process in which students acquire knowledge by creating products, analysing the properties and the functionality of their own products, and finally derive motivation to improve these products. Critical thinking asks us not to teach products of science and technology and their applications, but to teach the creative process of their development. To implement this approach, we use the historical method, allowing the students to learn by productive failures in the process of searching for a solution.

The concept presented in this paper approaches four competences. The students solve tasks for problem instances derived from common algorithmic problems, such as the *knapsack problem*, *vertex cover* or *dominating set*. Step by step, the student is asked to verify, find, evaluate, and finally optimize solutions for given problem instances. The tasks are designed for the third and fourth grade of elementary school implemented in an online learning environment based on the web framework *Vue.js* and were then exhaustively tested.

## 2 Didactic Goals

In this section, we present the didactic goals of the learning environment. The general didactic concept targeted aims to equip students to be able to encounter new problem formulations and learn how to solve them on their own. According to *Bloom's Taxonomy*, there are different levels of complexity depending on the cognitive dimensions of a task [1]. Here, four main competences are approached, which are based on the "Problem Solving and Algorithm Curriculum" that combines concepts such as constructivism and critical thinking with the hierarchy of *Bloom's revised taxonomy* [5]. These four approached competences are as follows:

1. To understand the abstract problem description and give proof of comprehension by classifying solution candidates into feasible and unfeasible solutions.

2. To find a solution for a given problem instance.

3. To find several different solutions for a given problem instance.

4. To apply criteria to evaluate and compare solutions, and to search for optimal solutions.

The task levels for each task set are designed accordingly: Starting with simple challenges that are classification tasks – the curriculum is designed such that the difficulty of the assignments improves gradually. Step by step, the students are taught the targeted algorithmic skills, similar to the classification informatics tasks of the *Bebras challenge on informatics* [4]. In this way, active learning and self-improvement are promoted. The students should learn to think in a way that enables them to approach new challenges and assignments with sophisticated steps and to solve them.

The tasks are designed such that the different levels of a curriculum look very similar. This creates recognition value, i.e., the user is familiarized with the problem and thus knows the general concept of the task when approaching the next difficulty level. Additionally, with multiple feasible solutions for problem instances, the student's creativity is required and challenged. The students learn to solve similar problems with yet different solutions or, in other words, learn how to apply algorithms. Random task instances further minimize the possibility of cheating or copying solutions. The students are compelled to find ways to solve the problems on their own, since there are no external sources or databases to help them. We will now link each task level with a competence or learning objective and describe each one in more detail.

## Level 1: Understand Problems and Verify Solution Candidates

The first competence level approaches the student's capability to interpret a given problem instance description. He or she has to decide whether a given solution candidate is a feasible solution or not. In doing so, the student proves the competence of correct interpretation. A solution candidate is feasible if and only if it fully meets the task specifications. The tasks are modelled as decision problems, where the user must answer at least one question regarding the correctness of the proposition. The student is required to interpret the proposition of the problem instance correctly, fully understand the criteria, and then evaluate the given solution proposition accordingly. Furthermore, in some cases, the students learn to justify why a specific problem instance is or is not feasible.

## Level 2: Find Solutions

At this competence level, the students are confronted with tasks for which a feasible solution must be found without any further help. The learning objective here is to work independently to find ways of dealing with a new challenge. This is more difficult than just verifying a solution candidate, since the student's creativity to come up with a new solution is required. The only condition a solution must fulfil is that it is valid. The quality of the solution according to a criterion is not taken into consideration here. All solutions are considered of equal value. One exercise type at this level has an intermediate version: the user is asked to complete a partial specification of a solution or instructed to find a solution on his or her own. In this case, however, the computer has already taken several steps towards a feasible solution candidate, so the number of valid solutions decreases significantly. If this eliminates some easy solutions, the difficulty level may change in comparison to an original level 2 task instance.

### Level 3: List Multiple Solutions

The third competence level approaches the competence to find multiple solutions that differ from each other. In contrast to the previous level, it is less useful to use a mere brute-force search and more effective to develop a strategy to solve the problem. Students could later learn to use decision trees to systematically list all solutions as a new competence.

### Level 4: Evaluate Solutions and Determine Optimal Solutions

Here, finding optimal solutions represents the highest competence level. It requires the student to evaluate different solutions with respect to a given criterion. This enables the student to compare the solutions and select an optimal one. Given the target group, it is evident that the number of possible solutions must be kept small such that a task instance can actually be solved. The variety of solutions must be small enough to enable the student to list all the different solutions, and to choose one of the best.

## 3 Test in Schools

In a final stage of the project, the learning environment was deployed on a test website and tested in various schools and grades. In this way, feedback for minor adaptions and improvements and empirical test data could be gathered. The test procedure was designed and arranged to include students from a wide range of origin, background, and academic performance to enable genuine and representative feedback to be gathered. In the following subsection, the test procedure will be examined and explained. The statistical results of the live testing feedback will then be presented in detail.

### 3.1 Test Extent

#### 3.1.1 Test Locations and Levels

The testing phase was carried out in various schools and in different grades. In total, a variety of third and fourth grade classes as well as groups of especially talented students from second to eight grade completed the testing procedure. Further, online access was distributed to teachers from various places who agreed to conduct the live testing and to provide feedback. In total, over a hundred people tested the platform. The feedback was collected, examined, and evaluated in order to improve the learning environment.

### 3.1.2 Test Procedure

The test procedure was structured in the following way: First, information about the project, instructions, and guidelines was announced. Then, each participating student was provided with a computer or convertible, and was instructed to solve problem instances within a given time slot for each different task type. The students were encouraged to try to understand the exercise on their own and use the tutorial in case of problems. At the end of each time slot, the exercise type was changed. At the end of the test, a feedback round was conducted, and each student had to fill out an online feedback form to enable statistical data to be gathered.

## 3.2 Empirical Test Results

The feedback form to collect statistical data was designed the following way: First, general statistics were gathered such that the feedbacks could be categorized by grade, age, and school. The feedback form contained five main paragraphs, where the interviewee had to describe the perception of the learning environment. The paragraphs included questions on how interesting the tasks were, how much fun it was to solve them, whether the tasks were easy to understand, whether the tutorial was useful, and whether the number of exercises was satisfying. Some statistics are shown below to illustrate the feedback. For simplicity's and brevity's sake, only overall results are presented. The data includes representative feedback from about 70 students, randomly distributed in terms of origin, background, and academic performance.

First, the platform was evaluated in terms of understandability and intuitiveness. The participating students were asked how long it took them to understand the exercises, and whether they spent much time figuring out what the exact task description was. The statistical result of the survey is visualized in Figure 1. Over 90 percent of all students considered the tasks to be intuitive and easy to understand. Only a small percentage did not understand the tasks and had to ask someone else for help. These results line up with the design requirement that the learning environment should provide an intuitive and self-explanatory interface.

Further, how often the users did not understand tasks and then opened the tutorial was measured. Note that the tutorial contains a text and tutorial video that can be opened if the task is unclear. On the one hand, the data collected provides insight into the approaches users choose to handle new tasks. On the other hand, it yields information on the intuitiveness of each task. The result of this test is visualized in Figure 2 and aligns with the statistics laid out previously on intuitiveness of the tasks (see Figure 1).

Additionally, observations have shown that most of the students immediately tried to solve the task by testing and trying things out by themselves instead of

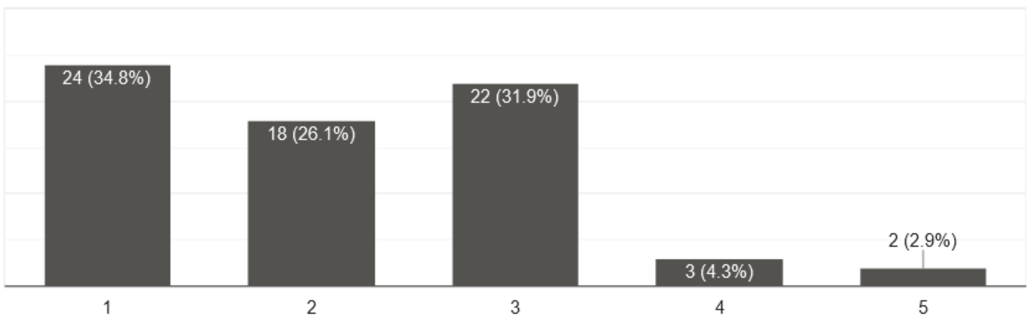

Figure 1: Diagram representing the task intuitiveness, linearly scaled from 1 to 5. 1 corresponds to 'very intuitive' and 5 to 'very unintuitive'.

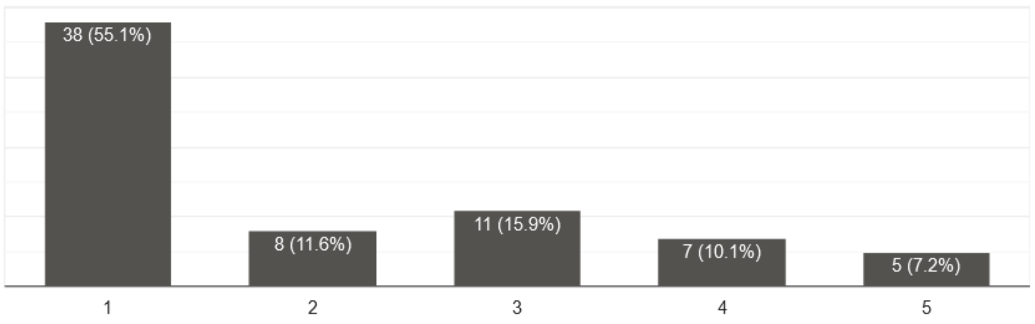

Figure 2: Diagram representing the consultations of an additional tutorial, linearly scaled from 1 to 5. 1 corresponds to 'not very often' and 5 to 'very often'.

reading the instructions. According to the collected data, more than half of the students tested did not even look at the tutorial once, and less than 20 percent opened it regularly when solving different tasks. This conclusion lines up with the feedback from testing staff, who noticed that some students did not even read the introductory sentence for each exercise, and immediately started clicking on the screen until they began to understand how to solve the respective task.

Next, statistics on user satisfaction were gathered. The user was asked whether he or she liked the exercises and was content with the way the tasks were constructed. The result of this survey is visualized in Figure 3 and matches the written feedback of the students. An overwhelming majority of the users indicated that they were pleased with the learning environment. Over 80 percent of the students liked the tasks or liked the tasks a lot. Only a small percentage did not like all the exercises (see leftmost bars in Figure 3). The written feedback analysis has shown that challenges that were perceived to be either too easy or too hard resulted in a low fun factor for a given task. These results agree with academic performance

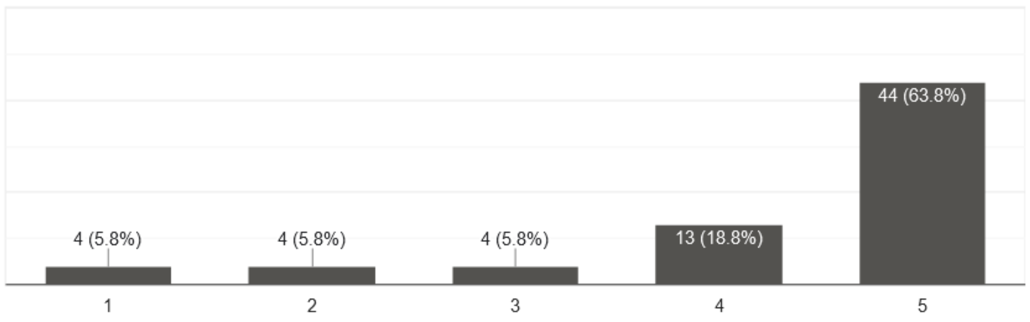

Figure 3: Diagram representing the perceived fun factor, linearly scaled from 1 to 5. 1 corresponds to ‘not so much fun’ and 5 to ‘much fun’.

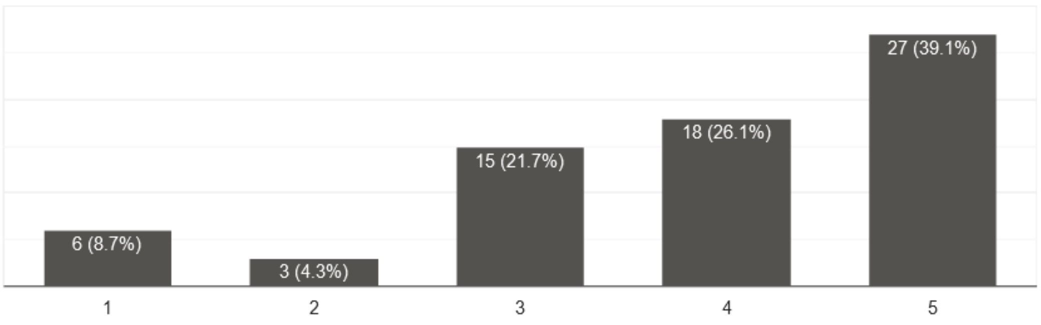

Figure 4: Diagram representing the perceived interest attached to the tasks, linearly scaled from 1 to 5. 1 corresponds to ‘not very interesting’ and 5 to ‘very interesting’.

statistics, which follow a normal distribution.

Further, an analysis on the attractiveness of the learning environment was performed. The tested students had to evaluate how interesting and exciting the tasks were. The visualization of this data set can be found in Figure 4. The result was rather positive, since nearly 90 percent of the students thought the tasks were at least more or less interesting, and overall nearly 40 percent rated the exercises as highly fascinating.

Subsequently, specific data to measure the difficulty of the tasks was collected. The users were asked how difficult the exercises were to solve, and whether the problems were challenging. Also, the time it took to solve single task instances was measured. Figure 5 gives a rough overview of the feedback data for this section. The results showed a normal distribution, as was expected. A small percentage of students was able to handle the tasks effortlessly, others were overwhelmed by the different task sets; and most students were seriously challenged. However, the statistics gathered suggest that slightly more students found the tasks to be on

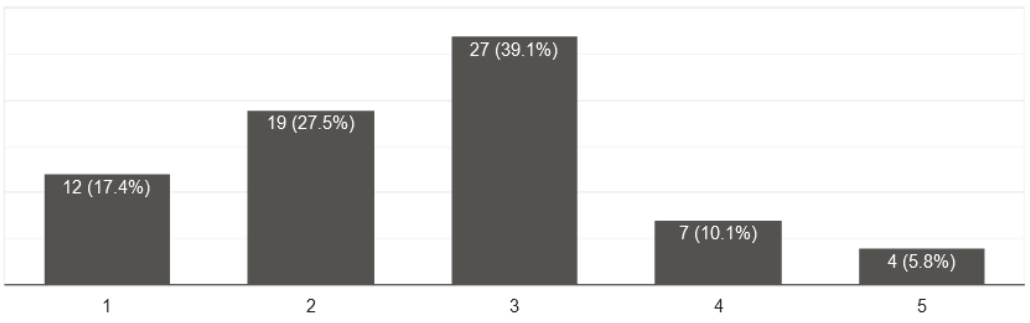

Figure 5: Diagram representing the perceived task difficulty, linearly scaled from 1 to 5. 1 corresponds to 'easy' and 5 to 'hard'.

the easy side within the normal distribution.

In the final evaluation section, the students were asked whether the number of exercises was sufficient. The result of this poll is visualized in Figure 6. About three out of four students were content with the variety of tasks.

Lastly, written feedback on specific questions was collected. The students were asked what they did or did not like about the platform, and what improvements they desired. The results of this questionnaire do not deviate from the previously presented data. In total, two out of three were highly positive: Over 20

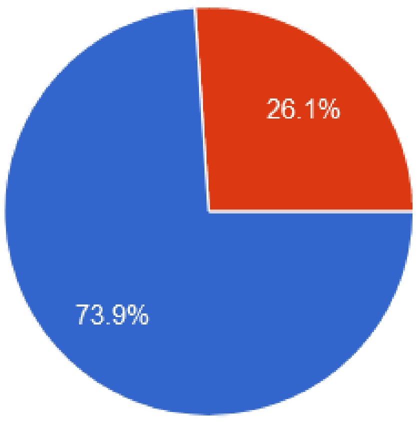

Figure 6: Diagram representing the perceived task variety. The fraction coloured in blue stands for 'good task variety', and the one coloured in red stands for 'rather small task variety'.

percent of those questioned explicitly remarked that they were pleased with the learning platform as a whole; 17 percent explicitly mentioned a specific task set on *Vertex Cover* as exceptionally interesting; 15 percent called the *Dominating Set* tasks well-designed. The tasks concerning the *Knapsack Problem* were mentioned in about 10 percent of the feedback forms. Furthermore, about 30 percent of the students explicitly indicated that they had no suggestions for improvements, while only a few individuals elaborated on ideas for enhancing the learning experience. Further, the feedback data supported the optimization of the learning environment in terms of user experience and performance. Linguistic vagueness could be clarified in several task descriptions, and additional explanations were added to improve understandability. Also, the environment could be further technically enhanced as some program parts were revised as a result of specific feedback. This concludes the feedback from the testing phase.

## 4 Conclusion and Discussion

The development of technical and didactic ways of teaching students is vital to ensure a sustainable education in the 21st century. Our contribution is to characterize an online learning environment that features diverse tasks that support the development of algorithmic thinking. The empirical test results indicate that the developed concept and derived tasks are user-friendly, intuitive, and interesting for the targeted student group. So far, the presented didactic concept has succeeded in the conducted tests. Everyone is invited to join the process by contributing and providing further feedback. Especially, new ideas for task concepts and test approaches are greatly appreciated.

### Acknowledgements

First of all, I would like to thank Juraj Hromkovič for the opportunity to write this paper, the uncomplicated procedure, and feedback during the process. Also, I would like to express my gratitude to Dennis Komm, Regula Lacher, Giovanni Serafini, and Esther Baumgartner for the ongoing support and constructive feedback throughout the proceedings.

## References

[1] Lorin W. Anderson and David R. Krathwohl: *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives.* 2nd edition. Longman, New York 2001.

[2] Tim Bell: Establishing a nationwide CS curriculum in New Zealand high schools: providing students, teachers, and parents with a better understanding of computer science and programming. *Communications of the ACM*, 57(2):28–30, 2014.

[3] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley: Computer science unplugged: school students doing real computing without computers. *New Zealand journal of Applied Computing and Information Technology* 13(1):20–29, 2009.

[4] Valentina Dagienė, Juraj Hromkovič, and Regula Lacher: A two-dimensional classification model for the Bebras tasks on informatics based simultaneously on subfields and competencies. In *Proc. of the 13th International Conference on Informatics in School (ISSEP 2020)*, LNCS 12518, pages 42–54, Springer, 2020.

[5] Valentina Dagienė, Juraj Hromkovič, and Regula Lacher: Designing informatics curriculum for K–12 education: from concepts to implementations. *Informatics in Education*, 20(3):7–15, 2021.

[6] Peter J. Denning and Paul Rosenbloom: The profession of IT – computing: the fourth great domain of science. *Communications of the ACM*, 52(9):27–29, 2009.

[7] Jens Gallenbacher: *Abenteuer Informatik: IT zum Anfassen für alle von 9 bis 99 – vom Navi bis Social Media* 4th edition. Springer, 2017.

[8] David Harel: *Algorithms – The Spirit of Computing.* Addison-Wesley, 1987.

[9] Heinz Hofer, Juraj Hromkovič, Regula Lacher, Pascal Lütscher, and Urs Wildeisen: *einfach Informatik 3/4: Programmieren und Rätsel lösen* (Textbook for Students). 1st edition. Klett und Balmer AG, 2021.

[10] Heinz Hofer, Juraj Hromkovič, Regula Lacher, Pascal Lütscher, and Urs Wildeisen: *einfach Informatik 3/4: Programmieren und Rätsel lösen* (Textbook for Teachers). 1st edition. Klett und Balmer AG, 2021.

[11] Juraj Hromkovič: *Algorithmic Adventures: From Knowledge to Magic.* Springer, 2009.

[12] Juraj Hromkovič and Regula Lacher: The computer science way of thinking in human history and consequences for the design of computer science curricula. In *Proc. of the 10th International Conference on Informatics in School (ISSEP 2017)*, LNCS 10696, pages 3–11, Springer, 2017.

[13] Dario Näpfer: *An Interactive Learning Environment.* `https://bit.ly/einfachInformatik34`. Department of Computer Science. ETH Zurich, 2021. Last accessed 19 Jul 2021.

[14] Kristen Nygaard: Program development as a social activity. In *Proc. of the IFIP 10th World Computer Congress*, pages 189–198, 1986.

[15] Giovanni Serafini: *The Benefits of Computer Science Education in Primary School: Promoting Algorithmic Thinking and Mathematical Problem Solving.* Manuscript in preparation. Department of Computer Science. ETH Zurich, 2022.