TEACHING COMPUTATIONAL THINKING WITH <COLETTE/>

Rebecca S. Stäter, Tim Läufer and Matthias Ludwig
Goethe University Frankfurt, Germany

Abstract. Computational Thinking (CT) is a necessary skillset to navigate and participate in our digitalized world. It is, thus, imperative to teach this skillset in school. Because CT is not automatically acquired simply by using digital tools, but it needs to be taught deliberately. However, a lack of easy-to-use teaching material and teacher training has been shown to hinder the adoption of CT-specific education in the classroom. Here we show a low-threshold approach to teaching CT using the <colette/> web portal and app. The web portal provides a convenient way for teachers to create tasks and the app allows students to display, solve, and review the tasks. Teachers create custom paths by adapting the predefined task family templates, each addressing a specific CT-skill. In this way, <colette/> facilitates teaching CT and integrating it into various school subjects.

Key words: Computational thinking, digital teaching, generic tasks.

INTRODUCTION

The ongoing digitalization of modern society makes it necessary to be skilled in using computers daily in various situations, for example in the workplace, but also increasingly for mundane matters. Surprisingly, the skills needed to employ computers (and computing devices) efficiently are not acquired by solely using them without guidance but need to be taught deliberately. In 2006 Wing coined the term Computational Thinking (CT) meaning this skillset needed in a digitalized world.

"Computational Thinking represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use." (Wing, 2006) At its core, this attitude or skillset is independent from technology and has been debated which skills or competencies are related to it and if they are disjoint from each other. Bocconi et al. (2016) have worked through numerous papers and found that the skillset most referred to as CT consists of: Abstraction, Decomposition, Generalization, Algorithmic Thinking, Automation, and Debugging.

Abstraction is the skill used to see a real-life object and discard unnecessary details to then replace this real-world object with a fitting model. Even the best fitting model might make it necessary to further decompose (Decomposition) the given problem and maybe one recognizes patterns in similar problems to then find a general way of solving them (Generalization). This process of solving the problem can involve clearly defined steps (Algorithmic Thinking) which are ideally understandable not only by humans but also by machines. If there is an algorithm which is understandable by a machine, we can exploit that fact by using the machine e.g., for repetitive tasks (Automation). Lastly, with the skill Debugging one can find errors in a given code as well as understand how an algorithm works by mentally going through each of its steps.

This CT skillset is not only necessary to deal with computers but it's a set of skills that can be generally applied to solving problems. All of these skills are not only skills necessary if one deals with computers, but they are a set of generally applicable skills while solving problems.

Stäter, R. S., Läufer, T., & Ludwig, M. (2023). Teaching Computational Thinking with <colette/>. In M. Ludwig, S. Barlovits, A. Caldeira, & A. Moura (Eds.), *Research On STEM Education in the Digital Age. Proceedings of the ROSEDA Conference* (pp. 123–130). WTM. https://doi.org/10.37626/GA9783959872522.0.15

(Wing, 2006). As CT-skills are not only applicable for coding, it is possible to integrate them also in other school subjects than Computer Science. Thus, CT is most of the times integrated in the curricula in one of the three following ways (Bocconi et al., 2016): As a cross-curricular theme, as part of a separate subject or within other subjects. For all the three possibilities there is a necessity for qualified teachers and for easy-to-use, inexpensive material to teach CT (van Borkulo et al., 2020).

In this paper, we show how the <colette/> app and web portal will allow teachers to easily integrate CT-specific contents into their lessons. As automated learning environment, <colette/> will be pre-populated with various Task Families (acting as blueprints for the specific tasks) and teacher training materials and it allows students to independently solve and review the chosen tasks. In what follows, we first introduce the <colette/> project before showcasing the workflows of creating CT-specific tasks via the web portal and solving and reviewing the tasks in the smartphone app. Afterwards, we address the general concept of Task Families, and discuss the specific ideas for the two already available Task Families.

TEACHING COMPUTATIONAL THINKING VIA THE <COLETTE/> WEB PORTAL AND APP

Project Goals of <colette/>

The research-project <colette/> conducted by a consortium of six partners all across Europe (Germany, Austria, the Netherlands, France, Slovakia). The aim of the project is to create a web portal, an app, didactical content and a short-term curriculum for teacher trainings, in order to facilitate integrating CT into preexisting lessons. In addition, serving as authoring tool, the web portal includes teacher material such as introductions to CT and guides on how to approach the topic in class, while the smartphone app gives the students direct access to material, hints and the feedback using their own smartphone ("bring your own device"-approach).

Creation of Paths with Tasks in the Web Portal.

To visualize how <colette/> allows teachers to integrate CT into lessons, Figure 1 shows a screenshot of the <colette/> web portal, as it appears when creating a new learning path (Roth, 2015). In an initial step, the teacher assigns the path a name (e.g. "Fun with Drones"), sets the target group (e.g. "upper secondary") and gives a description. Afterwards teachers can start adding various tasks to the path.

Adding tasks starts by selecting a task family (Figure 1, step 1), in this example the screenshots are taken from the Task Family *Building Cubes*. To date, we have made available two different task families, *Building Cubes* and the *Drone* (for further details see *Task Family: Building Cubes* and *Task Family: Drone*), with each task family acting as a sort of blueprint for the creation of specific tasks. In a later section, we'll expand further on the concept of task families, and present the two available task families in detail. After choosing a task family, the teacher is guided through the creation of a specific task. In the example of Figure 1, the teacher chose the task family *Building Cube,* and selected *Implementation* as assignment type ("*What should the student do?")* (Figure 1, step 2). In this assignment type, the student will create an algorithm to place cubes on a 3D grid to create a desired shape, the result will be

visualized in augmented reality (AR). In step 3 the teacher picks which shape will be implemented. In case of Figure 1 it is a *square pyramid*.

The next step is defining the settings of the pyramid. Here the teacher can choose the height of the pyramid and the location by defining a starting block through given (x,y,z) coordinates (Figure 1, step 4). The right-hand side of Figure 1 (step 4) shows a preview of the defined pyramid. Finally, step 5 allows to take a screenshot of the preview, change the problem definition and adjust or delete the hints. When the teacher completed all steps, they then can add the task to the path.

Once all tasks are created a 5-digit alphanumerical code is generated that identifies the path and can be shared with students to enter in the app.

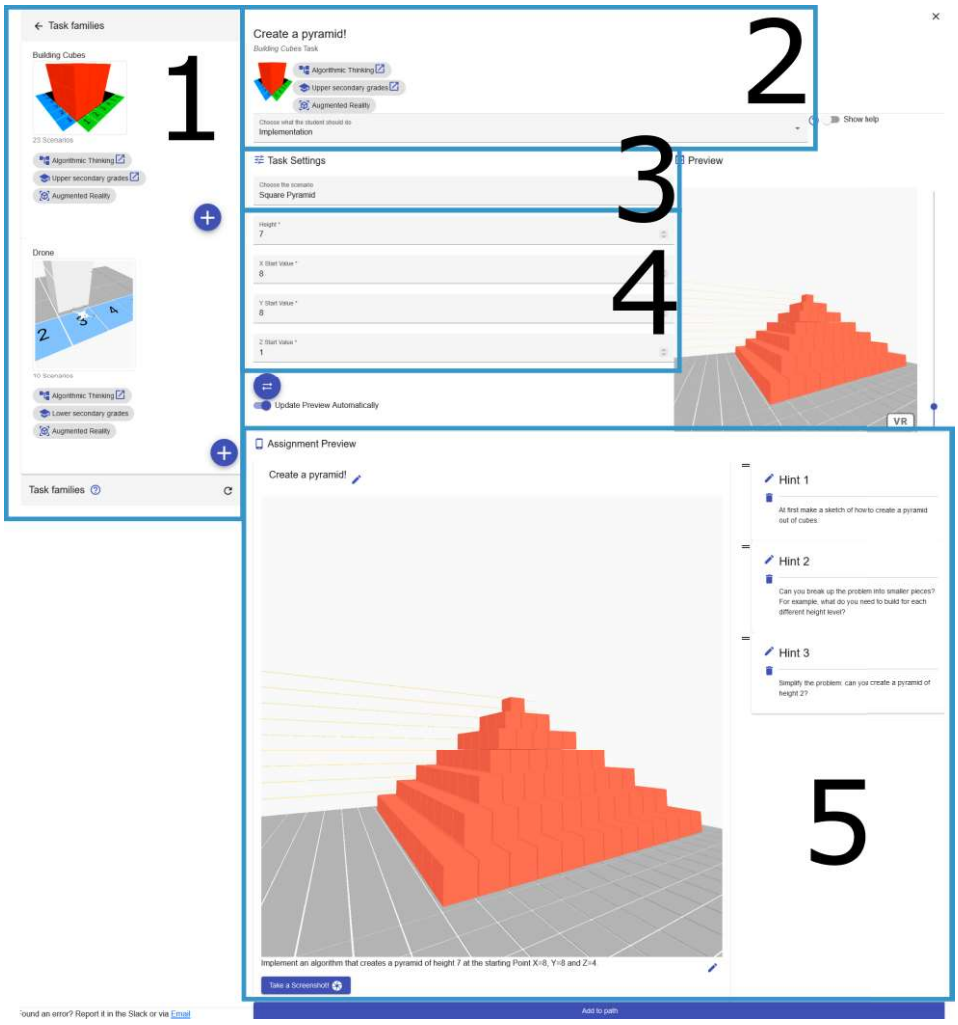

Figure 7: Creating a task in the <colette/> web portal (screenshot).

Figure 2 shows the workflow of a teacher in an abstract way. The teacher starts by selecting a Task Family and an assignment type (*"What should the student do?")* and then chooses a scenario and final settings. The problem definition, picture and hints can be edited as well. Note how the teacher's choice of the *"What should the student do"* (assignment type) corresponds to a specific *solution verification process* that the portal will employ. The

solution verification process allows the portal to automatically verify a given solution. After the settings have been set the sample solution and the expected result have been created. The expected result is in Figure 1 the actual pyramid structure.

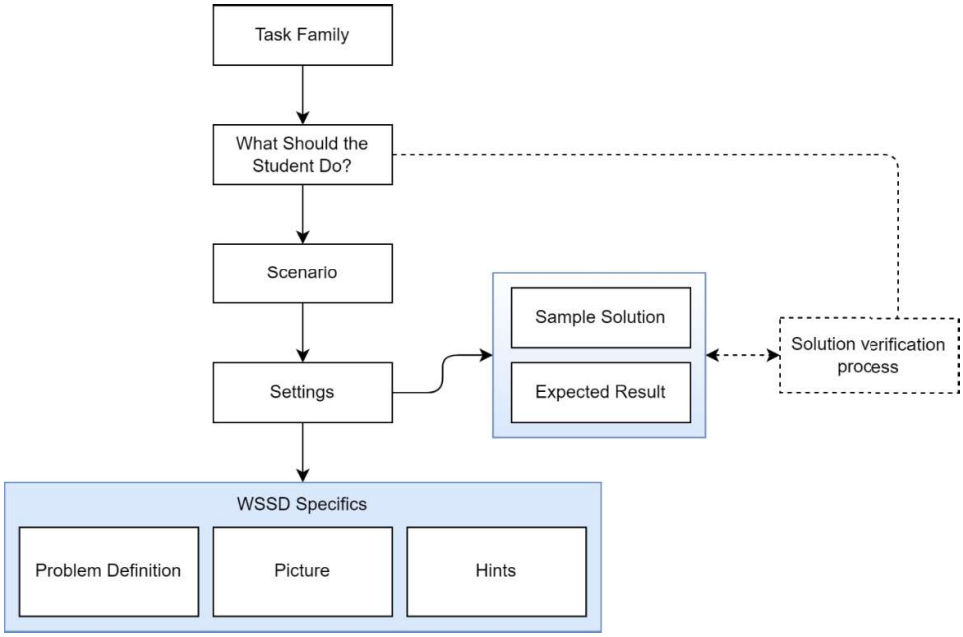

Figure 8: This flow chart shows the process of creating a task in the <colette/> web portal.

Viewing and Solving a Path in the <colette/> App

Once a learning path has been created by the teacher, the alphanumeric code of this path will be shared with the students. After starting the smartphone app, any previously added paths are visible in a list. New paths are added by tapping the small plus sign on the upper right-hand side of the screen (see Figure 3a). Entering the path code into the shown dialog (Figure 3b), the students can view the paths general information (Figure 3c) and start the path by clicking the start button in the lower part of the screen, in Figure 3 the student will work on a task from the Task Family *Drone*. The blue retry-button at the right resets progress in this path. The red "remove" button (shown by a -) on the left-hand side of the paths symbol removes the path from the list of saved paths (Figure 3c). If this was chosen the path can be readded with the code.

In Figure 3d we see a list of all the tasks in the path, the first one being the only one available to us at this moment. The others are not yet unlocked but can be started once all previous tasks are solved or skipped. Figure 3 goes on to show a task from the *"Drone"* task family, where students use block-based programming for the flight path of an augmented reality (AR) drone. Tapping on one task opens the screen seen in Figure 3e where we can see a row of categories for block-based programming expressions at the top. These can be opened and the specific expressions dragged down into the *Blockly* workspace (cf. Weintrop, 2019). Blocks that are no longer needed can be dragged into the trash bin in the lower right-hand side of the workspace to be removed. Below is the assignment given by the creator of the task, that can be collapsed to expand the workspace.

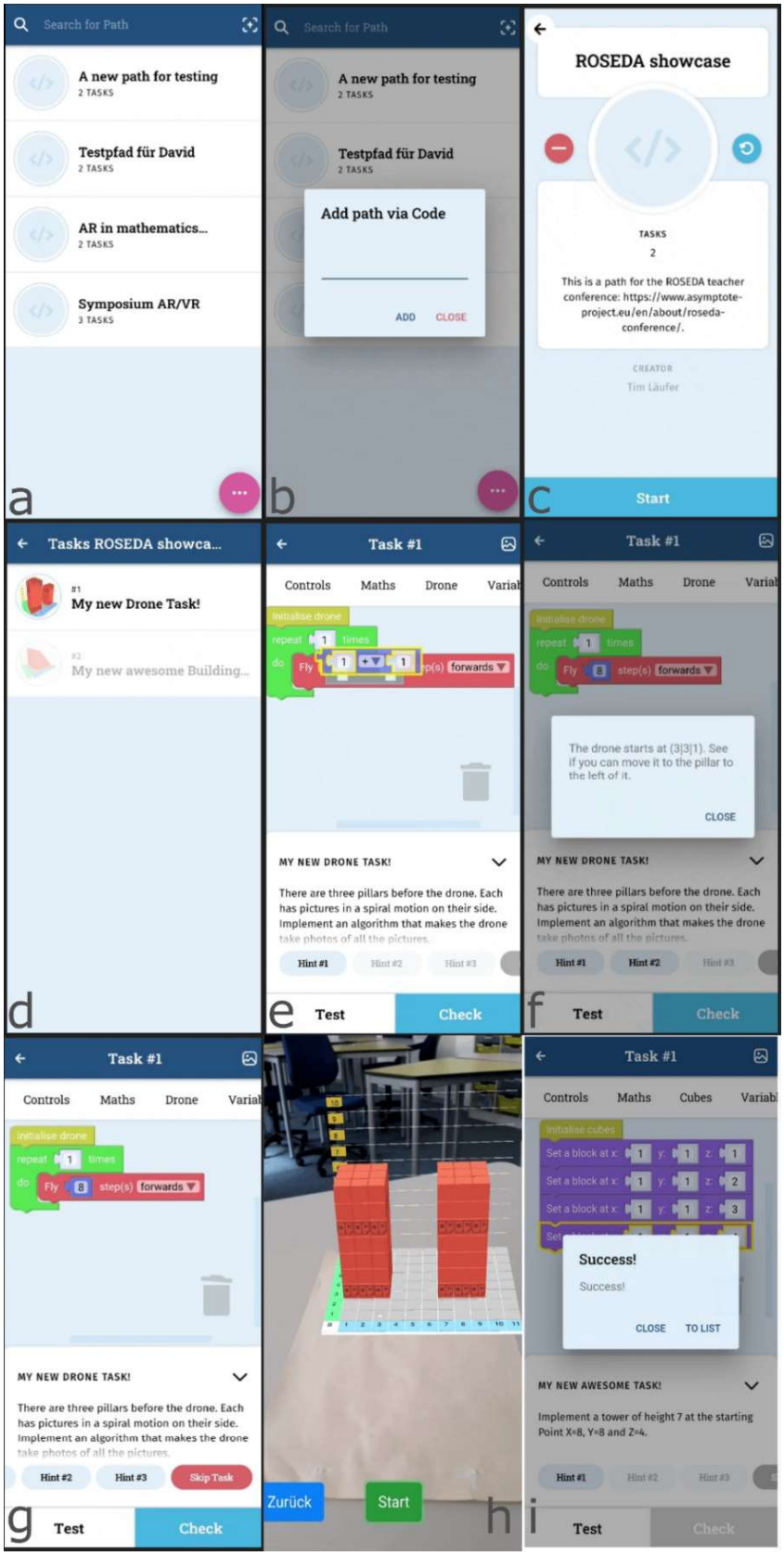

Figure 9: Viewing, solving and reviewing a task with the <colette/> smartphone app (screenshots).

The tiered hints can be tapped to open them (Figure 3f). When all hints have been consecutively opened the task can be skipped in case a student has trouble solving this task (Figure 3g). The button "Test" opens the AR preview (Figure 3h), where the solution of the programmed algorithm is displayed. There students can validate their answers before tapping the button "Check" to have the solution verified by the app (Figure 3i).

TASK FAMILIES AS EXTENDED GENERIC TASKS

Having seen the workflows of creating and solving a task, we now address the underlying concept of task families. In an earlier publication (Milicic et al., 2020), we introduced the concept of "generic tasks", meaning a simple structure for creating up to five different tasks from a common set of algorithm, wanted outcome and a description. By combining only those three components one can easily (and in automated fashion) derive five different tasks: implementing or analyzing a code, finding the error in a given code solving Parson's problem, as well as Parson's problem with distractors. A Generic Task itself, thus, cannot be assigned to a student. It is rather a building plan of a specific task.

Despite the variances in the task design, those Generic Tasks facilitate the process to provide material for different CT-skills as well. In our previous study, 14 experts concluded that these Generic Tasks address abstraction, algorithmic thinking, automation and debugging, thus supporting four out of the six CT-skills.

The concept of the Task Families generalizes the idea of the Generic Tasks. Task Families are a blueprint with which one can create several Generic Tasks which all have some settings in common. This is the foundation for the task editing process in the <colette/> web portal. We now discuss the two already available task families *Building Cubes* and *Drone* and show how Task Families expand the concept of Generic Tasks.

Task Family: Building Cubes

The Task Family *Building Cubes* allows the creation of tasks that revolve around 3D-shapes built on an augmented reality (AR) checkerboard, that can be viewed through the smartphone app directly on the students' desk using the AR marker. The shapes consist of individual cubes that are placed in their position via an algorithm that is editable with the visual code *Blockly.* For larger and more complex shapes, this building algorithm typically requires various loops and conditions.

From the perspective of Generic Tasks, a given shape (e.g. a pyramid), the building algorithm, and the problem definition can be seen as the three ingredients from which the five generic tasks can be derived. Currently, <colette/> provides the assignment type *Implementation*, in which students are asked to create an algorithm, and the existing algorithm is only used in the back-end to automatically verify the solution. In further iterations, we'll provide the other Generic Tasks (analyzing the code, debugging, Parson's problem).

The Task Family extends this concept of Generic Tasks by giving the choice of multiple figures to create. Teachers can choose between a cube, a table, a car and more. This means that the Task Family is a collection of Generic Tasks.

The assignment type *Implementation* addresses the CT-skill *Algorithmic Thinking,* while other assignment types will address other CT-skills (Milicic et al., 2020). Apart from CT-specific skills, we note that working with the augmented reality (AR) environment also addresses spatial recognition skills (Boon, 2003). Evidently, 2-dimensional screens (or paper) can only show 3D-objects as 2D-projections. Mentally translating 2D- into 3D-shapes (and vice versa) is a skill that needs to be trained and viewing and interacting with the shapes in the AR-view helps develop it. A student can walk around the programmed object (by walking around the AR-marker) and looking at it more closely and see it as if it were on their desk.

Task Family: Drone

The task family *Drone* is like *Building Cubes* in that it works with 3D-shapes placed on an AR-checkerboard, and the algorithm of interest is editable via *Blockly*. However, in the *Drone* task family, the algorithm is not used to create the 3D-shape (which is fixed here), but rather to direct the flight of a (virtual) drone, and have it perform tasks such as taking pictures from the viewpoint of a certain position.

The 3D-shapes are set up by the web portal in a generator, with available shapes comprising structures such as a wall with a window. In the previously shown example (Figure 3h) two pillars are placed on the checkerboard, and markers are placed in an upward spiraling pattern on the sides of those pillars. In the assignment type *Implementation*, the student programs the flight path of the drone in such a way that all the markers are being photographed subsequently by the drone.

Importantly, the movement of the drone is directly displayed in the AR view as animation, as well as the photos it takes. In comparison to the *Building Cubes*, we thus expect the *Implementation* task of the *Drone* task family to have a stronger process-character, because the students can investigate their algorithm sequentially and localize the error while the drone performs the actions. Therefore, *Debugging* is strongly addressed, as well as *Algorithmic Thinking*.

CONCLUSION AND OUTLOOK

In this paper, we presented our vision for <colette/>, an automated learning environment designed to facilitate teaching CT in class. We showed how teachers can use the web portal to create tasks by customizing a set of predefined Task Families that specifically target CT-skills, and how students can use the smartphone app to view, solve and review these tasks. We discussed how the concept of Task Families allows to (automatically) create variations of tasks from a common set of parameters, helping teachers to create effective material quickliy.

In addition to the two presented Task Families *Building Cubes* and *Drone*, we have designed more Task Families that will be available in the public release of <colette/> (first public release scheduled for 2023). Additional Task Families are designed for unplugged lessons and will be available in the handbook of <colette/>.

We have presented the assignment type *Implementation*, in which students themselves create the code of interest with *Blockly.* While this assignment type mostly addresses the CT-

skill *Algorithmic Thinking*, the public release of <colette/> will make available more assignment types (thus following the concept of Generic Tasks) and thereby addressing a whole spectrum of CT-skills.

With the addition of more Task Families, we will also emphasize that <colette/> is not exclusively designed for Mathematics and Computer Science classes but can be integrated in various school subjects. With this, we aim to reflect the general applicability of CT-skills, which are essential to navigate, participate, and understand an increasingly digitalized world.

Acknowledgment

The project is (partially) funded by the ERASMUS+ grant program of the European Union under grant no. 2020-1-DE03-KA201-077363. Neither the European Commission nor the project's national funding agency PAD are responsible for the content or liable for any losses or damage resulting of the use of these resources.

References

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). *Developing Computational Thinking in Compulsory Education: Implications for Policy and Practice.* Publications Office of the European Union.

Boon, P. (2003). Meetkunde op de computer. *Tijdschrift voor nascholing en onderzoek van het reken-wiskundeonderwijs, 22*(1), 17–26.

Milicic, G., Wetzel, S., & Ludwig, M. (2020). Generic tasks for algorithms. *Future Internet*, *12*(9), 152.

Roth, J. (2015). Lernpfade – Definition, Gestaltungskriterien und Unterrichtseinsatz. In J. Roth, E. Süss-Stepancik, & H. Wiesner (Eds.), *Medienvielfalt im Mathematikunterricht* (pp. 3–25). Springer.

van Borkulo, S. P., Kallia, M., Drijvers, P., Barendsen, E., & Tolboom, J. (2020, July 11–18). *Computational Practices in Mathematics Education: Experts' Opinions*. 14th International Congress on Mathematical Education (ICME-14), Shanghai, China.

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22–25.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35.