Solving Everyday Challenges in a Computational Way of Thinking

Bernhard Standl(✉)

Institute of Software Technology and Interactive Systems, TU Wien, Vienna, Austria
bernhard.standl@ifs.tuwien.ac.at

Abstract. Over a decade, computational thinking (CT) has been in the focus of educators and researchers in computer science. During this period of time, the term has been developed in different ways, reaching from a fundamental idea for finding a definition what problem-solving in computer science is about to a very particular view that CT is a required skill to code software applications. This paper presents results of the Fulbright project *coThink - Computational Thinking* carried out at the Missouri State University in Springfield, MO, USA which was based on the research question: How can CT be utilized with computer science algorithms for challenging real-life situations? As a result of a literature review, a CT five-step problem-solving process aimed at improving students' awareness to handle everyday life situations was identified. It was further integrated in classroom lessons, where it was applied at four student groups and evaluated mixing qualitative (analysis of worksheets) and quantitative methods (questionnaire) at a sample size of n = 75. Results showed that students frequently discovered a good approximation to solve real-life challenges following computer science algorithms but we also came to the conclusion to revise our problem-solving process.

Keywords: Computational-thinking · Problem-solving · Classroom research

1 Introduction

Seymour Papert initially introduced the term computational thinking (*CT*) in programming computers with LOGO for K-12 in 1980 in his book *Mindstorms* [11], but with problem-solving in mathematics in his mind. On the other hand Papert's problem-solving processes were influenced by Poyla's work from the 1950 s in *How to solve it* [12] who described how mathematical problems can be systematically approached and solved by students. The current movement in CT was initiated when Wing introduced the term again in 2006 [17]. Despite some similarities between Wing's and Papert's definition, Wing emphasized on problem-solving and Papert focussed on using CT to *forge ideas* [11]. Still, both didn't necessarily involve a computer in the process [6,9]. Considering this, we

V. Dagiene and A. Hellas (Eds.): ISSEP 2017, LNCS 10696, pp. 180–191, 2017.
https://doi.org/10.1007/978-3-319-71483-7_15

identified in recent literature, CT is for the most part either understood as umbrella term for learning coding concepts as e.g. loops, variables, recursions or as a term describing a problem-solving approach as used in computer science. Just as Nickerson et al. argued in [10]: *Definitions of CT vary*, there is, however, only little consensus in a definition for CT and so far a common understanding can only be identified in Wing's view [19]: *CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.* Barr and Stephenson argued in [2] referring to Wing [17], that CT includes seeking algorithmic approaches. This underlines that algorithmic thinking is part of CT and a term for composing different approaches, techniques, and systematics used in computer science for solving problems. Considering Lee, who argued in [8] that algorithmic thinking is a part of CT and further Barr, where CT is defined as the *seeking algorithmic approaches to problem domains* [2] in this paper we are focusing on the algorithmic part of CT. Even if the term CT is popular these days, so broad and deep it has been developed in history over decades. However, Tedre and Denning further described in [14] that some CT strategies can be applied also in different contexts, but they also claim, that *such transfer has never been substantiated.* Taking Yadav's thought, that CT can be integrated for everyday life challenges into account [21], this paper introduces an approach for combining computer science algorithms with real-life applications (as e.g. finding the shortest path to the school) within the theoretical framework of CT aimed at increasing the students' algorithmic problem-solving awareness by addressing these questions: *What is a suitable step-by-step computational-thinking problem-solving process for finding algorithms in real-life applications? How can such process put in practice classroom teaching at high-school?* The paper is structured as follows: First we are discussing related literature and propose a framework for a CT problem-solving processes for real-life applications, then we are presenting the integration of the process in classroom practice followed by an evaluation in practice and close with a conclusion and an outlook for further work.

2 Background

2.1 Algorithms for Everyday Life Applications

Algorithms in computer science have been developed for computers but they also can address human questions of everyday life. Our motivation for finding real-life applications is based on Wing's point of view, that techniques of computer science can support solving problems in other areas of life [17] where students identify learning as more authentic and has a strong relation to their own life. A further reason for connecting algorithms to human living was also, that this approach allows students to focus on the process of problem-solving beyond coding in a programming language. Inspired by popular ideas by Christian and Griffiths in *Algorithms to Live By* in [4] and scientifically confirmed in *Algorithms Unplugged* by Vöcking et al. [15] we selected five algorithms for an application in classroom practice: binary search, bin packing, minimal spanning tree, topological sort, and shortest path. Supposing, that the algorithms itself are known to the reader of this paper, we are presenting here the algorithms with only a brief description of the idea for an application with everyday life problems: **Binary Search:** When searching a CD or a book among many in a shelf, students develop a search strategy to find the item quickly. This leads to search algorithms and to the binary search algorithm in particular. **Bin Packing:** By asking how moving boxes can be filled up effectively with things from closets and shelves, students should find out strategies to master this systematically. The algorithms Next Fit and First Fit will describe the solution in computer science. **Minimal Spanning Tree:** The challenge is to connect seven islands with each other in building bridges. The distance between each bridge is known. Students have to find out the shortest possible connection to each island connecting the mainland. Solutions will lead to Prim's and Kruskal's algorithm. **Topological Sort:** Students write down a to-do list for a day and try to bring this list into order. By doing this, dependencies and preconditions will lead to the aimed algorithm for topological sorting that is used at finding one possible topological order. **Shortest Path:** Students work on a street map and try to identify the shortest path from home to school. The idea of selecting these algorithms was, to show in examples that solutions from complex problems in computer science can be easily transferred to challenge real-life situations as well.

2.2 Our View on Computational Thinking

Beyond all differences in definitions for CT, we identified across almost all contributions, that CT is about an approach for problem-solving in a way, that is used in computer science identified as algorithm. Wing already has stated in her refined definition of CT in [18], that CT is an approach for solving problems that draws on concepts fundamental to computing. Further, Aho described in [1] that the term CT includes *algorithm-design and problem-solving techniques that can be used to solve common problems arising in computing.* This is similar to Wolfram Math World's definition of the term *algorithm* itself as a specific set of instructions for carrying out a procedure or solving a problem. The question arises, if there is an additional value in introducing CT or if CT is just a vague and fashionable term paraphrasing procedures already included in the term *algorithm*? In fact, narrowing the term CT to designing algorithms would be off its underlying idea. Selby et al. explained for instance in [13] CT should include the ability to think in abstractions, in terms of decomposition, algorithmically, in terms of evaluation, and the ability to think in generalizations. Moreover, Xu et al. described in [20] CT as a universal, not limited to computers, fundamental, not only for experts, provides mental tools for thinking processes, has rich contents and is connected to any concept of computer science, and poses deep scientific problems in providing a framework to ask meaningful scientific questions. As Yadav et. al. reminded in [21,22] that Wing's initial paper [17] points out that CT involves three key elements *Algorithms, Abstraction, and Automation*, the term CT has been grown since then to a variety of interpretations. For

Lee et al. [8] CT involves defining, understanding, and abstraction. Barr et al. suggested in [2] CT involves the design of solutions, implementation of designs, testing, running analysing, reflecting, abstraction, creativity, and group problem solving. Grover and Pea [7] stated that CT should include among others abstraction, information processing, structured problem-solving decomposition as modularization, iterative recursive thinking, and efficiency. Again, for Lee et al. [8] CT involves defining, understanding, and solving problems, reasoning at multiple levels of abstraction, understanding and applying automation, and analysing the appropriateness of the abstractions made. In reference to Voogt et al. [16] who suggest, that *...we should not try to give an ultimate definition of CT, but rather try to find similarities and relationships in the discussions about CT,* outcomes of our literature review resulted in a problem-solving approach which goes beyond the skills computer literacy and understanding technology and underlines the need for developing CT skills [5]. Beyond the actual problem-solving process, Barr et al. suggested in [2] also a collection of dispositions a problem-solver should hold, which will be included in our process bellow. Those are confidence, persistence, handle ambiguity, deal with open-ended problems and communication skills in team-work. From teaching practice experiences, we know that the teacher's awareness of such dispositions at the students can be crucial for the success of learning and added them as additional part of the process.

2.3 Problem-Solving Approach for Real-Life Challenges

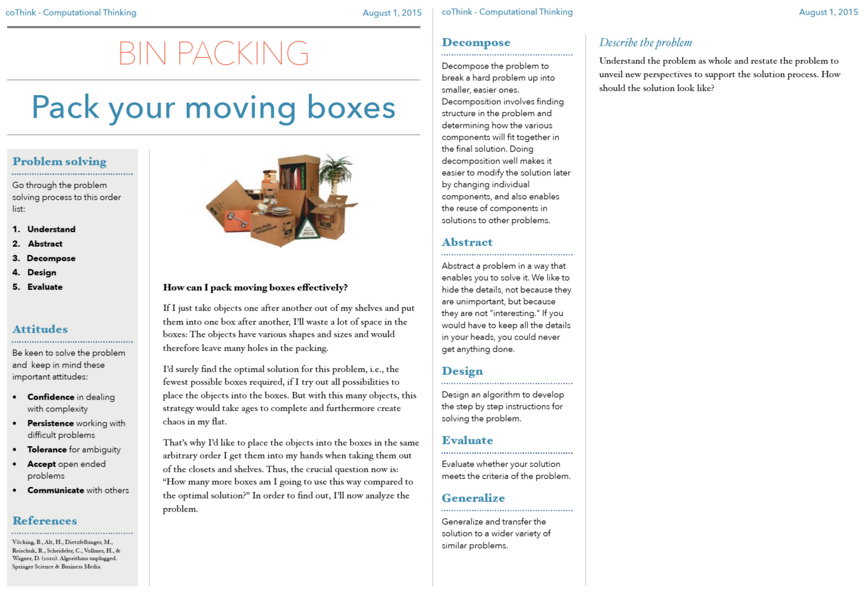
The goal of our CT problem-solving process is, to improve the students' awareness to be able to take advantage of CT problem-solving techniques to solve real-life problems. Therefore, we suggest an arranged step-by-step process for solving a problem. The process includes besides elements as abstraction, decomposition, and design of algorithm also understanding the problem at the beginning of the process and testing the solution at the end of it. **1: Understand** the problem as whole and restate the problem to unveil new perspectives to support the solution process. Also, state clearly what should be achieved with the solution. This part derives from Papert's initial ideas on CT [11], referencing to Poyla's first problem-solving step *First, you have to understand the problem* in *How to solve it* [12]. **2: Abstract** a problem in a way that helps to solve it. If we had to keep all the details in our heads, we could never get anything done. As we have described above, abstraction is mentioned as a keystone of CT and Grover et al. identified it as a core element, which differs CT from other types of thinking [7]. **3: Decompose** the problem to break a hard problem up into smaller, easier ones. Decomposition involves finding structure in the problem and determining how the various components will fit together in the final solution. Doing decomposition well makes it easier to modify the solution later by changing individual components, and also enables the reuse of components in solutions to other problems. This was already suggested by Wing in [17] as modularizing a problem *to make it tractable.* We see the parts decomposing and abstraction are key steps for preparing the actual problem-solving process. Abstraction is a step, where parts found will be removed or reduced in its representation and decomposing is a process, where the remaining parts will be separated and clustered. **4: Design** an algorithm to develop the step-by-step instructions for solving the problem. Start from what already is known and work outward from there. Make a plan how to approach to solve the problem. Selby et al. [13] relate to Wing [17] and further describe this process as a heuristic reasoning to devise a solution. **5: Test** the algorithm whether a solution meets the criteria as testing and debugging as used in software development and in the context of CT suggested by Brennan et al. in [3] to smoothen not working parts of the solution. This process has been planned by us so that it can be carried out in a linear fashion step by step for solving the problem. As mentioned above, we enhanced this five-step problem-solving approach, with these dispositions, as it was suggested by Barr et al. in [2]: **Confidence** in dealing with complexity, **Persistence** in working with difficult problems, **Tolerance** for ambiguity, The ability to deal with **open ended problems**, The ability to **communicate** and work with others to achieve a common goal or solution. Taking this approach for a CT problem solving approach into account, in the next section we will describe, how we integrated it into classroom practice.

3 Method

In order to proof the concept, we carried out our interventions at two high-schools with two classes each. The participants (n = 75) were students between the ages of 15 and 17 from four different high-school classes at two different schools (Greenwood Laboratory School and Willard High School located in Springfield, MO, USA). As the access to high school classes was limited to 2×2 h, we designed our lessons with a strong emphasis on your five-step problem-solving approach. Our goal was to find out, if everyday life problems can be approached systematically using techniques from computational thinking. Therefore, we planned four lessons on two days with the overall intention to let students find a solution to a problem heuristically followed by a reflection compared to the actual algorithm of computer-science. Students not only practiced their problem-solving competencies but also learned how computer science concepts can be useful and meaningful for their own life. We accompanied the classroom activities with mixed methods using a pre-/post-questionnaire and an analysis of the students' work sheets. Even though, both schools offer computer science classes with instruction on using a computer and standard software, students of both schools received no prior computer science education with a focus on computational algorithmic thinking on a regular basis. Hence, we presumed that all participating students had similar pre-experiences with algorithmic problem-solving processes. Due to this fact, we identified all students across both schools as one entity for analysing the data. It should also be mentioned that we did not include any control groups for organizational reasons and because of the research design, which was primarily aimed at evaluating our problem-solving concept. We carried out two separate lessons (2 h each) on two different days with each of the four student groups at two schools, the first lesson introduced CT and our problem-solving approach and in the second lesson students worked on problems individually and in groups collaboratively. In order to provide students a guideline through the problem-solving process, our worksheets were distributed with a clear description of the problem and a path through the process which was the same as stated above from understanding the problem for generalizing the solution.

3.1 Instruments

We designed two research instruments for evaluating our problem-solving process: in order to track the students' problem-solving strategies, we designed worksheets. The worksheets had five sections, divided into the five-step problem solving process. In Fig. 1, the first two pages of a worksheet to bin-packing are depicted. The front page describes the problem to the student and recalls the five-step problem-solving process and the supportive attitudes in the column on left hand side. On the right-hand side, an exemplary part of the work-sheets shows, that students also had to sketch for each problem-solving step their ideas as mentioned above: describe the problem, abstract the problem, decompose the problem, Design the algorithm, test the solution. In order to identify differences in students attitudes towards problem-solving we distributed pre-/post-questionnaire. The questionnaire included 20 items, where each of Barr's dispositions [2] has been assigned to four questions using a Likert-Scale from 1 'strongly disagree' to 6 'strongly agree'.



coThink - Computational Thinking August 1, 2015

BIN PACKING

Pack your moving boxes

Problem solving

Go through the problem solving process to this order list:

1. **Understand**
2. **Abstract**
3. **Decompose**
4. **Design**
5. **Evaluate**

Attitudes

Be keen to solve the problem and keep in mind these important attitudes:

- **Confidence** in dealing with complexity
- **Persistence** working with difficult problems
- **Tolerance** for ambiguity
- **Accept** open ended problems
- **Communicate** with others

References

Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., & Wagner, D. (2010). Algorithms unplugged. Springer Science & Business Media.

How can I pack moving boxes effectively?

If I just take objects one after another out of my shelves and put them into one box after another, I'll waste a lot of space in the boxes: The objects have various shapes and sizes and would therefore leave many holes in the packing.

I'd surely find the optimal solution for this problem, i.e., the fewest possible boxes required, if I try out all possibilities to place the objects into the boxes. But with this many objects, this strategy would take ages to complete and furthermore create chaos in my flat.

That's why I'd like to place the objects into the boxes in the same arbitrary order I get them into my hands when taking them out of the closets and shelves. Thus, the crucial question now is: "How many more boxes am I going to use this way compared to the optimal solution?" In order to find out, I'll now analyze the problem.

coThink - Computational Thinking August 1, 2015

Decompose

Decompose the problem to break a hard problem up into smaller, easier ones. Decomposition involves finding structure in the problem and determining how the various components will fit together in the final solution. Doing decomposition well makes it easier to modify the solution later by changing individual components, and also enables the reuse of components in solutions to other problems.

Abstract

Abstract a problem in a way that enables you to solve it. We like to hide the details, not because they are unimportant, but because they are not "interesting." If you would have to keep all the details in your heads, you could never get anything done.

Design

Design an algorithm to develop the step by step instructions for solving the problem.

Evaluate

Evaluate whether your solution meets the criteria of the problem.

Generalize

Generalize and transfer the solution to a wider variety of similar problems.

Describe the problem

Understand the problem as whole and restate the problem to unveil new perspectives to support the solution process. How should the solution look like?

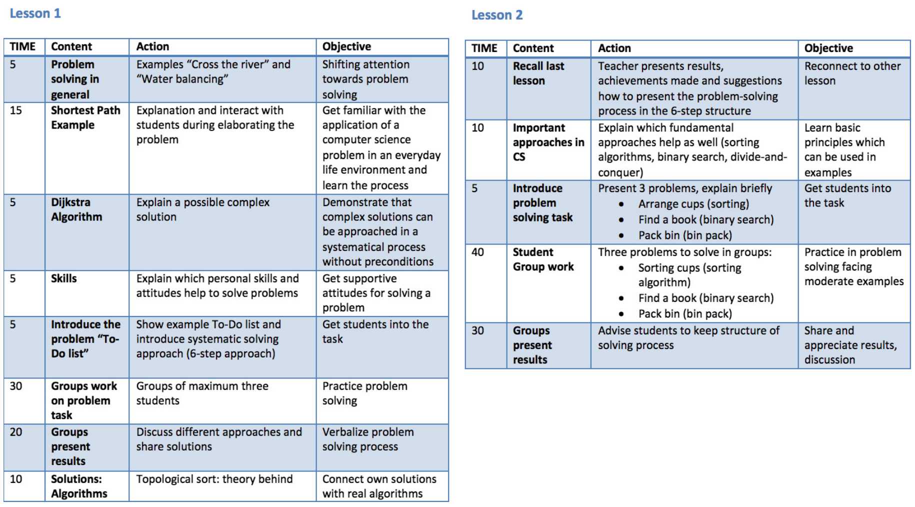
Fig. 1. Worksheet for the task "Pack your moving boxes"

3.2 Analysis

The analysis of the worksheets was a process of interpretation, where we evaluated our linear approach of solving the problem in going through each of the five problem-solving steps. We checked in which parts of the worksheets students were accepted as useful during the problem-solving process by reading through the worksheets and identified, which parts were not edited by the students where it was difficult for students to use it for the problem-solving process. The analysis of the questionnaire was carried out with descriptive statistics and statistical significance tests.

3.3 Classroom Lessons

The lesson plan in Fig. 2 gives an overview on the classroom intervention of day one and is based on findings of our literature research and includes all five steps from understanding the problem to generalizing the solution.



Lesson 1

TIME	Content	Action	Objective
5	**Problem solving in general**	Examples "Cross the river" and "Water balancing"	Shifting attention towards problem solving
15	**Shortest Path Example**	Explanation and interact with students during elaborating the problem	Get familiar with the application of a computer science problem in an everyday life environment and learn the process
5	**Dijkstra Algorithm**	Explain a possible complex solution	Demonstrate that complex solutions can be approached in a systematical process without preconditions
5	**Skills**	Explain which personal skills and attitudes help to solve problems	Get supportive attitudes for solving a problem
5	**Introduce the problem "To-Do list"**	Show example To-Do list and introduce systematic solving approach (6-step approach)	Get students into the task
30	**Groups work on problem task**	Groups of maximum three students	Practice problem solving
20	**Groups present results**	Discuss different approaches and share solutions	Verbalize problem solving process
10	**Solutions: Algorithms**	Topological sort: theory behind	Connect own solutions with real algorithms

Lesson 2

TIME	Content	Action	Objective
10	**Recall last lesson**	Teacher presents results, achievements made and suggestions how to present the problem-solving process in the 6-step structure	Reconnect to other lesson
10	**Important approaches in CS**	Explain which fundamental approaches help as well (sorting algorithms, binary search, divide-and-conquer)	Learn basic principles which can be used in examples
5	**Introduce problem solving task**	Present 3 problems, explain briefly • Arrange cups (sorting) • Find a book (binary search) • Pack bin (bin pack)	Get students into the task
40	**Student Group work**	Three problems to solve in groups: • Sorting cups (sorting algorithm) • Find a book (binary search) • Pack bin (bin pack)	Practice in problem solving facing moderate examples
30	**Groups present results**	Advise students to keep structure of solving process	Share and appreciate results, discussion

Fig. 2. Lesson plan day 1 and 2

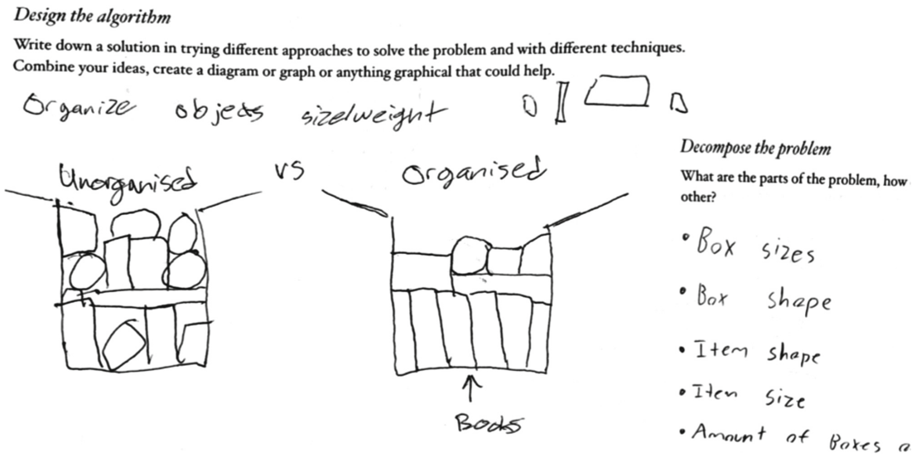
We started with an introductory lecture about problem-solving, CT and algorithms. Next the teacher approached a solution together with the whole student group for the shortest path problem by going through the five-step problem solving approach introduced above. Next, each student solved the example with the minimal spanning tree and shared the solution at the end of the hour together with others in group-work. It is important to note, that the students' attitudes required for a beneficial problem-solving process were mentioned explicitly by the teacher, who facilitated also the group-work for promoting the students' awareness for it. On the second day, the teacher first summarized insights from the first lesson and gave a lecture on some approaches, how computer scientists try to solve problems. This included basic sorting algorithms as well as general strategies as divide-and-conquer. Students subsequently worked in groups on more difficult problems as binary search, bin packing and the topological search as described above. Finally, group-work was followed with group presentations which included a discussion on connecting solutions with the *real* algorithms from computer science.

Example of application: In order to underline how our CT problem-solving process works for real-life challenges in detail, we give here an example based on the worksheet as depicted above. To each problem-solving section, we describe a possible student's answer which is based on Vocking et al. [15]. *Pack your moving boxes: How can I pack moving boxes effectively?* **Possible solution of a student:** By going through the problem-solving process, the students systematically approach the solution to the problem (very short version): **Describe the problem:** The challenge is to place my items into moving boxes in the most efficient way. **Abstract the problem:** To make it easier, I reduce the items to a geometrical representation that each part is described as block. So, a ball becomes for example a block with a edge length of its diameter. **Decompose the problem:** I have moving boxes with a certain size and items with different sizes and forms. **Design the algorithm:** First I tried different approaches in practice and extracted a general solution: 1. Sort the items by size, 2. Take the next available biggest item. 3. If the items fit into the box: Place the item into the box. Else: Open the next box. 4. Go to step 2. **Test the solution:** I organized a small shoe-box and some items and I evaluated my solution. It turned out, that my solution is not perfect yet as I need to many moving boxes. This example demonstrated, how the problem-solving process can be integrated for finding a solution to a problem from a real-life context by using a CT problem-solving approach leading to an algorithmic solution.

4 Results

4.1 Worksheet: Problem-Solving Process

Results showed, that students were successful in developing an approach to solutions for the problems. But they also experienced sometimes difficulties in dealing with the five-step process of solving a problem and were not always sure how each of the steps relates to the problem-solving process. For example, students were more engaged in describing the problem and designing the solution but they had difficulties to fill the worksheets for abstracting the problem, decomposing it and evaluating the solution separately. We found out, that the process of describing the problem already covered parts of the abstraction process. As in Fig. 3 it is shown, students have used different ways to solve the problem, some graphical, some using text. For instance, when students described the problem using their own words, some kind of abstraction has already taken place by most of the students. Next, the part of decomposing the problem in smaller parts apart from problem-solving was experienced as difficult and was rarely edited. Most of the time, decomposition was integrated into the design process or the evaluation process. These outcomes suggested us to reorder and reduce our problem-solving process at the worksheets in future to three steps: **1. Describe, decompose and abstract the problem**, **2. Design the algorithm**, and **3. Test the solution**.



Design the algorithm

Write down a solution in trying different approaches to solve the problem and with different techniques. Combine your ideas, create a diagram or graph or anything graphical that could help.

Organize objects size/weight

Decompose the problem

What are the parts of the problem, how other?

- Box sizes
- Box shape
- Item shape
- Item size
- Amount of Boxes a

Fig. 3. Snippets of filled student worksheets

4.2 Questionnaire: Students' Problem-Solving Attitudes

Even though, the structure of the worksheets was challenging for students, our intervention still had some impact on students' attitudes towards computational problem solving as the pre-/post-questionnaires suggest. The results of the questionnaires showed some differences at certain items, where some of them even differed statistically significant. The items of the diagram below can be clustered in four sections, where each cluster represents a field of dispositions a problem-solver should hold as defined above. These are **1–4:** Confidence in dealing with complexity, **5–8** Persistence in working with difficult problems, **9–12:** Tolerance for ambiguity, **13–16:** The ability to deal with open ended problems, **17–20:** The ability to communicate and work with others to achieve a common goal or solution. The questionnaire was identified as reliable as all two of the measures evidenced a good reliability as Cronbach's alpha ranged from 0.82 for the pre-test and 0.85 for post-test. As the results in Table 1 show, confidence (items 1–4) increased and item 1: "I feel anxious when I have to solve this kind of difficult problems." had also a significant difference in the pre-test scores (M = 3.55, SD = 1.51) and post-test scores (M = 4.20, SD = 1.26); t(140) = −2.77, p = 0.006. Furthermore we identified a significant difference in the pre-test scores (M = 3.55, SD = 1.42) and post-test scores (M = 4.20, SD = 1.12) at item 4: "I feel

Table 1. Aspects from students' attitudes towards problem solving. Scales from 1 'strongly disagree' to 6 'strongly agree' (n = 75)

	Item	∅ pre	∅ post
1	I feel anxious when I have to solve this kind of difficult problems	2.78	2.00
2	I doubt that I could solve this complex problem	2.26	1.73
3	I like to solve challenging problems like this one	3.74	3.55
4	I feel confident in solving this kind of difficult problems	3.70	3.91
5	If I could not solve this problem quickly, I would give up	2.30	2.18
6	I feel impatient when I'm working on this kind of challenging problems	2.87	3.09
7	When a problem like this challenges me, I am keen to solve it	4.17	4.09
8	I would work on this problem until I have found a solution	4.39	3.73
9	I rather prefer to work on problems where the solution is already obvious	3.04	3.09
10	There is possibly only one way to solve this problem	2.70	1.91
11	There are many roads that lead to the solution	4.13	4.64
12	This problem may has more than one solution	4.61	4.73
13	There exists a solution to this problem	5.04	5.55
14	I would be confused, if I wouldn't find a clear solution with this problem	3.78	3.82
15	There can be different approaches to solve this problem	4.78	5.00
16	The solution to this problem can look different from what I have expected	4.43	4.82
17	I don't think that someone could help me to solve this problem	2.13	2.27
18	If too many people are involved in solving this problem, it will not be done well	3.43	3.95
19	Teamwork is the key to solve this problem effectively	3.78	3.93
20	A solution to this problem could be better found together	4.26	3.27

confident in solving this kind of difficult problems.", conditions; t(140) = −3.00, p = 0.003. Items in the second cluster of persistence in the problem-solving process (items 5–8) decreased. The third cluster tolerance for ambiguity (items 9–12) increased and there was a significant difference in the pre-test scores (M = 4.06, SD = 1.14) and post-test scores (M = 4.74, SD = 0.89) at item 10: "There is possibly only one way to solve this problem.", conditions; t(140) = −3.59, p = 0.001. The fourth cluster on the ability to deal with open ended problems (items 13–16) increased and there was a significant difference in the pre-test scores (M = 4.94, SD = 1.51) and post-test scores (M = 5.34, SD = 1.26) at the variable "There exists a solution to this problem.", conditions; t(140) = −2.32, p = 0.022. The fifth cluster on teamwork (items 17–20) is inconsistent in terms of that students think that the solution to a problem can be found better together (item 20) but at the same time state that too many people involved in problem solving will have a negative impact (item 17). In a nutshell, results of the questionnaires suggest that students reduced their anxiety in solving difficult problems and increased their confidence. We further conclude, that students developed their understanding for the possibility of multiple solutions.

5 Discussion

This paper presented results of the project *coThink - Computational Thinking* which was aimed at identifying an approach for a step-by-step CT problem-solving process for real-life applications. The underlying idea was to break problem-solving techniques known from computer science down to support students' awareness for handling real-life challenges. Based on literature research a five-step CT problem-solving approach emerged which was enhanced with dispositions a problem-solver should hold as suggested by Barr et al. in [2]. We designed lessons and material which integrated our ideas and concepts and applied it in practice at four student groups at two different schools. First experiences showed, that students were keen to develop solutions for problems in a real-life context and their approach mostly approximates an algorithmic description or even sometimes an existing algorithm of computer science. Results of the questionnaires evaluating students' attitudes for problem-solving showed, that our intervention had some positive impact, even if it is likely that students didn't adopt their dispositions but rather had an increased awareness which dispositions are required for CT problem-solving. Further work will also include the refinement of new worksheets, the specification of further algorithmic real-life examples and its application in classrooms. It turned out, that students had problems filling all sections of the worksheets and finding a description for each of the five problem-solving steps was perceived as confusing. In fact, only the sections *Understand the problem* and *Design the solution* were accepted and helpful in the process. Some of the students described their solution, which was supposed to part of section *Abstract the problem*, as part of *Describe the problem*. As a consequence, we further redesigned the worksheets and combined fields which will lead to three sections of the worksheets, as described above (1. Describe, abstract and decompose the problem - 2. Design the algorithm - 3. Test the solution) In essence, we reduced the five-step problem-solving approach to three steps in order to simplify the process for students to: understand - solve - analyse. In the future, we will continue refining the teaching material and the research setting for a more detailed analysis focusing more on the actual students' problem-solving process by adding for instance think-aloud interviews as research instrument to gain more detailed insights how our approach has impact on CT problem-solving process.

Acknowledgments. This research was supported by the Fulbright Visiting Scholar Program Austria sponsored by the U.S. Department of State. I would like to thank the Fulbright Office in Vienna, the Council for International Exchange of Scholars (CIES) and the Institute of International Education (IIE). I thank my colleagues at Missouri State University in Springfield, MO, USA who provided assistance in facilitating the research project. I would also thank the school authorities and teachers at Greenwood Laboratory School and Willard High School for support and assistance in organization of classroom lessons and in submitting the IRB proposal.

References

1. Aho, A.V.: Computation and computational thinking. Comput. J. **55**(7), 832–835 (2012)
2. Barr, V., Stephenson, C.: Bringing computational thinking to K-12. ACM Inroads **2**(1), 48 (2011)

3. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking, pp. 1–25 (2012)
4. Christian, B., Griffiths, T.: Algorithms to Live by: The Computer Science of Human Decisions. Henry Holt and Co., New York (2016)
5. Dagiene, V., Futschek, G.: Bebras, a contest to motivate students to study computer science and develop computational thinking. In: Proceedings of WCCE, pp. 139–141 (2013)
6. Dagienė, V., Sentance, S.: It's computational thinking! Bebras tasks in the curriculum. In: Brodnik, A., Tort, F. (eds.) ISSEP 2016. LNCS, vol. 9973, pp. 28–39. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46747-4_3
7. Grover, S., Pea, R.: Computational thinking in K12 a review of the state of the field. Educ. Researcher **42**(1), 38–43 (2013)
8. Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L.: Computational thinking for youth in practice. ACM Inroads **2**(1), 32 (2011)
9. Mannila, L., Dagiene, V., Mirolo, C., Settle, A.: Computational thinking in K-9 education, pp. 1–29 (2014)
10. Nickerson, H., Brand, C., Repenning, A.: Grounding computational thinking skill acquisition through contextualized instruction. In: Proceedings of ICER 2015, pp. 207–216 (2015)
11. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Cambridge (1980)
12. Polya, G.: How to Solve It, 2nd edn. Penguin Books Ltd., London (1957)
13. Selby, C.C., Woollard, J.: Computational thinking: the developing definition, pp. 5–8 (2013)
14. Tedre, M., Denning, P.J.: The long quest for computational thinking. In: Koli Calling Conference on Computing Education Research, pp. 120–129 (2016)
15. Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., Wagner, D. (eds.): Algorithms Unplugged. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15328-0
16. Voogt, J., Fisser, P., Good, J., et al.: Computational thinking in compulsory education: Towards an agenda for research and practice. Educ. Inform. Technol. **20**, 715–728 (2015). https://doi.org/10.1007/s10639-015-9412-6
17. Wing, J.M.: Computational thinking. Comm. ACM **49**(3), 33 (2006)
18. Wing, J.M.: Computational thinking and thinking about computing. Philos. Trans. Ser. A Math. Phys. Eng. Sci. **366**(1881), 3717–3725 (2008)
19. Wing, J.M.: Computational Thinking: What and Why? The Link - The Magazine of the Carnegie Mellon University School of Computer Science (2011)
20. Xu, Z.W., Tu, D.D.: Three new concepts of future computer science. J. Comput. Sci. Technol. **26**(4), 616–624 (2011)
21. Yadav, A., Hong, H., Stephenson, C.: Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. TechTrends, 1–4 (2016)
22. Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., Korb, J.T.: Introducing computational thinking in education courses. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, pp. 465–470. ACM (2011)