Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment

Brandon Rodriguez, Cyndi Rader, and Tracy Camp
Dept. of Electrical Engineering and Computer Science
Colorado School of Mines, Golden, CO, USA
brandonrrodriguez@gmail.com, crader@mines.edu, tcamp@mines.edu

ABSTRACT

Computer Science Unplugged activities have been shown to be successful in increasing student interest in computer science when used in outreach and after school events. There is less research available on adapting these extra-curricular activities for use in a classroom setting, where there are more students and the activities must support educational goals, not just changes in attitude. We describe our work in updating several existing CS Unplugged activities as well as introducing some new activities for use in an American middle school classroom. One challenge when using CS Unplugged activities is to determine what, if anything, students are learning. In this paper we detail one approach that links the updated activities to computational thinking skills, then incorporates worksheets where students illustrate their understanding.

CCS Concepts

•Applied computing → Interactive learning environments;

Keywords

CS Unplugged, Classroom Assessment, Computational Thinking

1. INTRODUCTION

In 2006, Jeannette Wing (Carnegie Mellon University) coined the term Computational Thinking (CT) as a way that humans conceptualize computable problems [12]. Computational thinking is a process for solving or interpreting unstructured problems such that a computer could output an answer. Computational thinking is important when there are many solutions that can lead to a correct answer, and where some solutions may offer a computational advantage when using a machine to calculate the result. Since computers are pervasive in our society, teaching CT concepts at the middle school level will give students tools for effectively solving a variety of problems in different disciplinary areas.

Computational thinking encompasses much more than learning how to program. Following several years of discussion within the CS education community, the five CT skills that are commonly accepted are [2]:

- Data Representation: the act of representing information so that it may be effectively processed by a computer.
- Decomposition: breaking a problem into several smaller problems that, when solved, will answer the original problem.
- Abstraction: generalizing a problem to see if techniques from similar problems can be used to solve the current task.
- Algorithmic Thinking: designing step-by-step processes or applying known algorithms to obtain a solution.
- Pattern Recognition: identifying trends or discovering the cause of patterns in data.

Now that skills have been identified, the next challenge is to determine how to teach and assess CT in the classroom. Computer Science Unplugged (CS Unplugged) activities are a set of lesson plans made available for free on the internet [3]. The aim of these lesson plans is to convey fundamental computer science concepts to students without any computer skills and to help bridge the gap for K-12 teachers who may not have a technical background but are expected to teach technical ideas. CS Unplugged activities are kinesthetic, engaging, and above all emphasize that computer science is about problem solving, and not synonymous with programming. A number of studies have explored the impact of CS Unplugged on students' attitudes [7, 9]. In this paper we are primarily concerned with what students are learning. We explore two questions. First, what CT skills might we expect students to acquire from various activities? And second, can we devise instruments that integrate into the activities so that the assessments are both engaging and provide insight into what students are learning? We provide results from deploying our modified versions of CS Unplugged activities in two American middle schools.

2. RELATED WORK

Numerous research projects have presented methods for teaching computer science without the use of computers or

ITiCSE '16, July 09-13, 2016, Arequipa, Peru

DOI: http://dx.doi.org/10.1145/2899415.2899465

programming languages. CS Unplugged and "computer science magic shows" have been successful examples of teaching computer science through highly engaging activities for students [6]. CS Unplugged activities have been adopted by a variety of educational outreach programs such as after-school workshops and summer camps [3], and have even been incorporated into the Exploring CS Curriculum [5].

The majority of the studies conducted using these types of activities have been concerned primarily with increasing interest in computer science, and not necessarily about using the lessons in classroom environments or assessing what the students are learning by completing the activities.

Renate Thies and Jan Vahrenhold from the Technical University of Dortmund in Germany investigated the suitability of CS Unplugged activities for use in a classroom (instead of an after-school program) by using the activities with a group of students. They used CS Unplugged activities to teach half the students, and used alternative tools for the other half of the students. Their findings showed CS Unplugged activities were equally effective in transferring knowledge as there was no significant difference in achievement between the group who learned with CS Unplugged activities and the group who learned with alternative materials [11]. Additionally, the researchers studied the impact of using CS Unplugged activities in different grade levels, and found that the activities had a significant positive impact when used with middle school classes. Thies and Vahrenhold have also mapped CS Unplugged lessons to Bloom's Taxonomy to determine what level of cognitive processes are prompted by various activities [10].

Thies and Vahrenhold's research examined all unmodified CS Unplugged activities. CS Unplugged activities were originally designed to be used in outreach scenarios, and therefore do not explicitly list learning objectives. The researchers' extrapolation of learning objectives suggests that the CS Unplugged curriculum lies in the lower end of the Bloom's Taxonomy spectrum. The authors noted that higher level learning objectives are needed for middle school audiences [11]. Thies and Vahrenhold's research is of particular significance because they bridge the gap between entertaining outreach programs and measurable student outcomes that can be used in a traditional classroom. The extensions and new activities our group has developed specifically address higher learning objectives in order to make CS Unplugged materials better suited for secondary education.

Quintin Cutts of the University of Glasgow detailed how group exercises in classroom environments can be just as effective as one-on-one tutors. Group work can also increase confidence and encourage students to become personally interested in the material [4]. Cutts' work supports what we have observed while deploying CS Unplugged activities. Namely, students prefer working in groups, and students who otherwise would not have asked questions to the teacher are comfortable asking their peers for help.

Lynn Lambert of Christopher Newport University published an article in 2009 that deployed pre- and post-surveys to evaluate CS Unplugged activities [7]. Lambert's results found students showed an increase in confidence in computing topics, but failed to gain knowledge about computing careers. The conclusions from Lambert's study were the basis for our development of career related and real world extensions and lecture material for CS Unplugged activities. See our career extensions on our website for details [1].

3. THE STUDY

The focus of our study is assessing the use of CS Unplugged in middle school classrooms. Based on interviews with teachers and pilot tests in after school settings, we selected a subset of the CS Unplugged activities that seemed age appropriate. Since classroom time is valuable, we limited our deployments to approximately six activities (i.e., there are more than six activities that are suitable for middle school, but we studied only six in detail). For each activity, we developed a detailed lesson plan that would fill a 50-minute class period. All of the lesson plans have a similar structure that includes whole class discussions, collaborative group work, and individual work. As part of each activity, students completed worksheets that were collected by the researchers. Unless otherwise noted, worksheets were completed individually by each student. To remain true to the spirit of CS Unplugged [8], we strove to incorporate game-like challenges or stories into the worksheets. This is also consistent with CT goals, which represent a process or approach to problem solving rather than a set of facts that can easily be assessed using, for example, multiple choice questions. We should note that CT is not currently required material for middle school students in our state, so the purpose of these assessments was for us to measure the impact of the activities, not for the teachers to assign a grade.

We partnered with two local middle schools to deploy and assess activities in 7^{th}-grade classrooms. Both schools place extra emphasis on science and technology. Six CS Unplugged activities were deployed to students as part of their normal classroom work during the school day. Our first deployment was during fall 2015, with approximately 130 students. The results from that deployment highlighted a few issues with some of the activities and/or assessments. A second deployment during spring 2016 showed improved results, but we again identified issues with a few of the activities (e.g., insufficient coverage of some topics) and assessments (e.g., confusing instructions on worksheets). Results are presented below from our third deployment, which also occurred during spring 2016.

Table 1 shows a mapping of the Unplugged activities to CT, along with a brief list of any modifications we made to the original activities. A description of these extensions is beyond the scope of this paper, but lesson plans and worksheets for all revised Unplugged activities, including four that are not mentioned in this paper, are available on our website [1].This paper will focus on six activities (Finite State Automata, Binary Numbers, Cryptology, Error Detection, Minimal Spanning Trees, and Searching) with worksheets that assess computational thinking. The worksheet from the Searching activity included pairs of students; all other worksheets were done individually. The number of students varies from 64 to 122 and will be reported for each activity.

4. METHOD

After the classroom deployment, a rubric was created for each worksheet. The rubrics provided guidelines on how to score worksheet answers as either *Proficient*, *Partially Proficient*, or *Unsatisfactory*. Every worksheet was individually scored by two researchers to ensure consensus. Disagreements on any score were resolved by having both researchers score the question together and then editing the rubrics to

Table 1: Classification of CS Unplugged activities and their CT skills.

Activity Name	CT Skills	Our Contributions
Binary Numbers	Data Representation Pattern Recognition Abstraction	Check Your Understanding worksheet Bit Ranges worksheet Binary Go Fish activity
Cryptology	Decomposition Pattern Recognition Abstraction Algorithmic Thinking	New activity using Caesar cipher Encode/Decode worksheet Surprise party activity
Finite State Automata	Data Representation Abstraction	Lesson plan using fruit vendor rather than Treasure Hunt Robot Dog worksheet Chores Robot worksheet
Parity Bits and Error Detection	Data Representation Decomposition Algorithmic Thinking	ASCII and Parity worksheet
Searching	Algorithmic Thinking	Raffle ticket activity Guess My Number lecture Dragons and Cows worksheet
Minimal Spanning Trees	Algorithmic Thinking	Halloween Candy worksheet

better document any edge cases. Student scores were used to determine whether there was evidence that students could apply the concepts from the activities and, in some cases, to identify revisions to the activities and/or assessments.

As mentioned, the worksheets were designed to be integrated into the activity, rather than a "quiz" to be completed after the activity is done. Worksheets also needed to be straightforward enough to be completed within the time frame of a typical 50-minute class period. One consequence of this constraint is that the worksheets do not ask students to provide reasoning or show how they arrived at an answer. In determining the proficiency levels, students who scored *Proficient* arrived at the correct answer, *Partially Proficient* arrived at an incorrect answer, but one which the evaluators could understand (i.e., student was on the right track but had a clear misconception or made a computational error), and *Unsatisfactory* included responses where the student did not attempt the problem or where the answer was blatantly incorrect.

For brevity, only the Finite State Automata activity will be discussed in detail. Results are presented, however, for all six activities.

5. FINITE STATE AUTOMATA (FSA)

The FSA lesson plan explores problems and situations that can be represented with a set of states, and the transitions required to move between states. In this activity, students begin by playing a game without any introduction to FSA terminology or concepts. In groups of four or five, one student is designated as a fruit vendor and the other students as fruit buyers. In the game, the fruit vendor sells apples and bananas, but may not always sell what he/she is asked for. The buyers are tasked with finding a sequence of fruit purchases that will result in the vendor selling three apples in a row.

After groups realize the pattern, the entire class reassembles. The syntax for states, transitions, start state and stop state are introduced on the whiteboard, and a FSA diagram of the fruit vendor's selling pattern is constructed (with fruits as possible states, and purchase requests as transitions). Vendors are then given new instructions (a different pattern), and the buyers are again tasked with receiving three apples in a row. This time, however, they can represent their findings as an FSA to assist in articulating the vendor's pattern. An optional third set of vendor instructions allows for flexibility in timing and provides an additional fun activity for students who quickly identified the second pattern.

Once students have finished the fruit vendor activity, they are given two worksheets to apply their new FSA knowledge to different problems. In the first worksheet, students are given a complete FSA diagram that depicts a robotic dog's various states. They are asked to evaluate transition paths to determine what state the robot dog would be in after completing the transitions. In the second worksheet, students are given a list of rules for a chores robot. The rules explain the robot's functions, and clearly identify the different states the robot can take, as well as the different transitions the robot recognizes. Students must then take the list of rules and convert it into a well formed FSA diagram.

Results from the FSA worksheets are given in Figure 1. Because student performance on the worksheets was generally good, the scale on each figure is from 50 to 100%. A total of 103 students completed worksheets for this lesson. Students generally did well, with over 65% achieving *Proficient* scores on all questions. The results indicate that students struggled most with using an FSA diagram to generate a sequence of transitions that end at a specified state (specifically, they were asked to identify two different paths that would result in the robot performing all the chores). However, they were able to evaluate given paths of transitions with much higher success.

6. RESULTS FOR THE REMAINING ACTIVITIES

Brief summaries of the results for the other five activities are included in the following sections. Each section lists the number of students who turned in worksheets for that activity. Mapping the activities to CT skills has allowed for

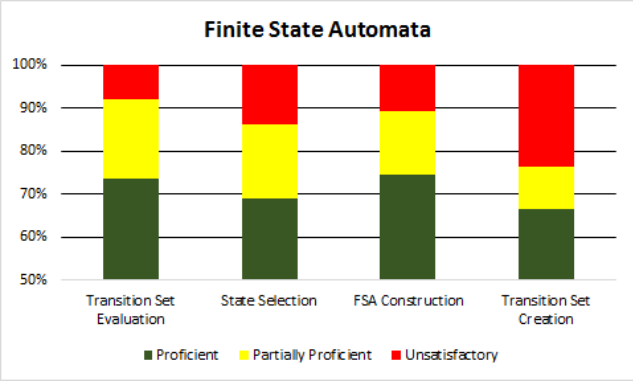

Figure 1: Results from our deployment of the FSA lesson plan in spring 2016.

our research team to measure what students are learning; thus, the results are more detailed than an attitude survey alone.

6.1 Binary Numbers

In the binary numbers activity, students learn that data is stored in computers as ones and zeros and they use binary flip cards to learn how to count in base two. As part of that discussion, students practice converting between number systems, and they use the flip cards to understand the range of values that can be represented by a 5-bit number. After the discussion, students complete a worksheet with six questions. Results are given in Figure 2. In general students performed well on this worksheet, with greater than 80% of the 79 students achieving *Proficient* scores on all questions.

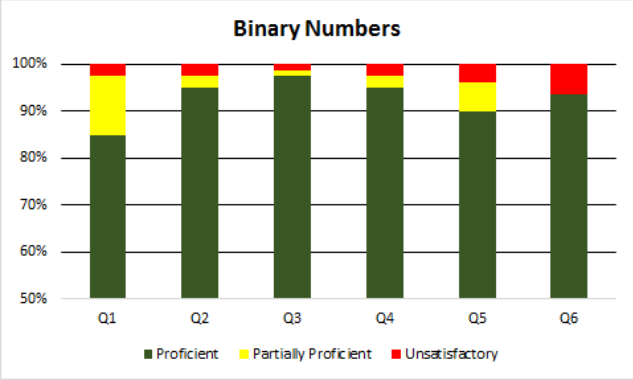

Figure 2: Results from our deployment of the Binary Numbers lesson plan in spring 2016.

The first question on the worksheet addresses the CT area of *Pattern Recognition*, as students are given a sequence of four binary numbers and asked what number would be next. The following two questions deal with *Data Representation*, as students are asked to convert a binary number to decimal and then a decimal number to binary. The final three questions incorporate several CT areas, including *Abstraction*, as students are asked about the largest number that can be represented with five bits (covered during whole class discussion) and then need to generalize this to three bits and six bits.

6.2 Cryptology

In the cryptology activity, students practice encryption and decryption using Caesar ciphers. As part of this activity, students create a cipher using a given key (e.g., offset by two letters) and then encode a message. Then, they analyze how many possible keys a Caesar cipher could have before using a given cipher to decode a message. To make the discussion more concrete, students explore why cryptology is important and what types of data they consider important to protect. The results from the cryptology assessments are shown in Figure 3, with 122 students being represented.

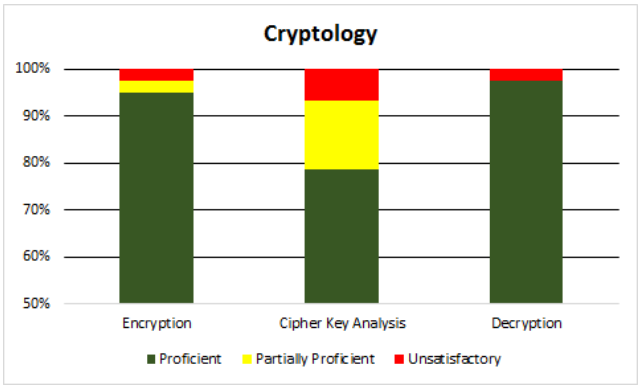

Figure 3: Results from our deployment of the Cryptology lesson plan in spring 2016.

Students showed overwhelming comfort with using Caesar ciphers to encrypt messages, with over 95% of students correctly encrypting a given message. They also did very well using a known Caesar cipher to decrypt an encrypted message, with 98% of students retrieving the original plaintext message. A few students struggled with using the Caesar cipher in reverse, and instead of decrypting the message, they re-encrypted it. A majority of students were also able to think about Caesar ciphers abstractly and determine how many possible cipher keys could exist using a simple cipher.

6.3 Parity Bits and Error Detection

The Parity Bits and Error Detection activity is an expanded version of the existing "Card Flip Magic" lesson from CS Unplugged [3]. Students start the lesson by seeing the magic trick. A classroom discussion introducing ASCII follows, and students learn that computers can sometimes make mistakes while transmitting data. Students then practice error detection on ASCII messages before realizing the same concept can be applied to explain the magic trick from the beginning of the class. Results from the ASCII worksheet are presented in Figure 4. A total of 118 students participated in this activity.

The results show that over 80% of students answered all questions at the Proficient level. In the first question, students were given a 7-bit ASCII chart and asked to convert binary patterns to letters. Almost all of the students were able to do this successfully. The second question asked students to compute the parity bit for four letters (they were given the first 22 letters as an example). In the final question, students were given a binary message and asked to identify the error. The questions related to parity were challenging for about 15% of the students.

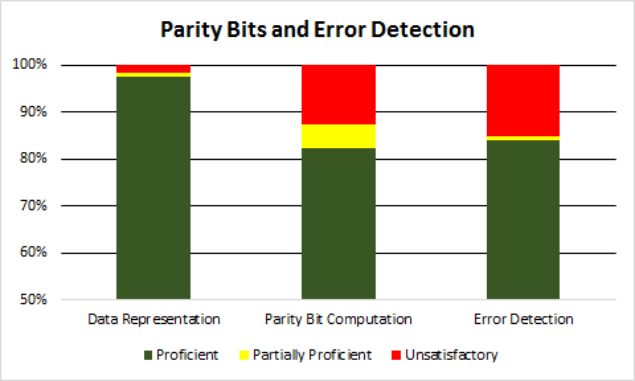

Figure 4: Results from our deployment of the Parity Bits and Error Detection lesson plan in spring 2016.

6.4 Minimal Spanning Trees

The Minimal Spanning Tree activity is an extension of the "Muddy City"activity from CS Unplugged [3]. Students first find a MST in the Muddy City worksheet using a brute force approach. After, students are introduced to Kruskal's algorithm for finding a minimal spanning tree (without any mention of proofs, graphs, edges, or nodes). They are then given another worksheet with a different graph to practice using Kruskal's algorithm. A classroom discussion helps identify where MSTs exist in the real world and how what they've learned can help solve certain types of problems. Finally (time permitting), students build their own graphs using pennies and toothpicks before rotating and finding the minimal spanning tree on their classmate's graph.

Results for the second MST worksheet (after students have learned Kruskal's algorithm) are given in Figure 5. Worksheets were collected from 121 students.

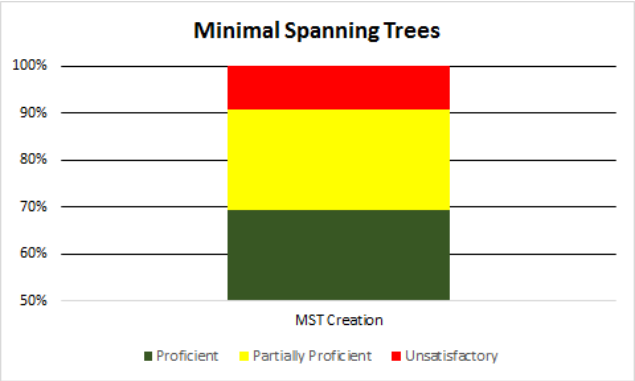

Figure 5: Results from our deployment of the Minimal Spanning Tree lesson plan in spring 2016.

A majority of students were able to correctly identify the minimal spanning tree by applying a given algorithm to the graph. When considering the students who scored *Partially Proficient* (where the MST was off by one or two edges), over 90% of students reached a solution.

6.5 Searching

The searching activity begins with a raffle ticket activity in which students try to locate a winning ticket. The optimal approach would be to use a binary search, but for this introduction students typically search in some random order. Next the ping pong demo, as described on the CS Unplugged website [3], is used to introduce binary search. An additional "guess my number" lecture is done via whole class discussion, to reinforce the concept. Finally students work in pairs on a worksheet in which students must search effectively (i.e., using binary search) to find the desired cow and save the entire herd from the attacking dragon. Figure 6 shows the results from the Dragons and Cows worksheet with 64 students.

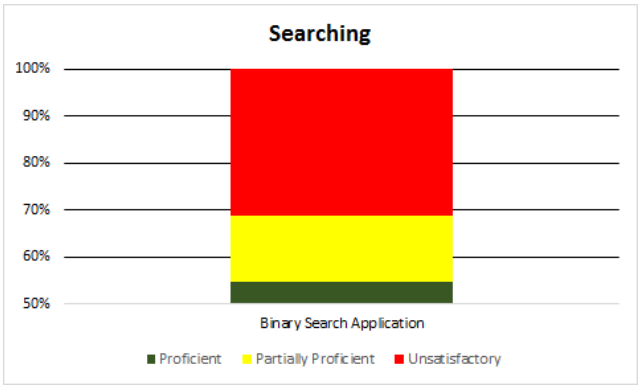

Figure 6: Results from our deployment of the Searching lesson plan in spring 2016.

As Figure 6 shows, only 53% of the students performed a correct binary search. Another 14% were on the right track but made a mistake in applying the algorithm. The remaining 31% did not apply a binary search. It's not clear whether the lack of performance is due to students not understanding binary search or due to the structure of the activity. The worksheet included 26 cows labeled A-Z, and students needed to find the specified cow (by weight) within five guesses. The students were not required to use binary search, but we hoped that they would choose to do so in order to find the cow quickly. The majority of students did choose one of the middle cows (M or N) for their first guess. But with only 12 or 13 cows remaining at that point, a number of students did not continue doing a strict binary search. Despite this, most students were able to locate the desired cow within five guesses. In this case, the activity was engaging, but not an adequate assessment of students' understanding.

7. EXTENDING CS UNPLUGGED

This paper presents results from six CS Unplugged activities. For the binary number activity, we followed the "Count the Dots" lesson plan from the CS Unplugged website [3] but developed our own worksheet to assess students' understanding. We also added an activity to reinforce the number of bits needed for a given number and a fun game (Go Fish) for students to practice binary conversions. The error detection activity uses the "Card Flip Magic" activity but with additional worksheets to more directly focus students' attention on parity bits. The lesson plan includes time for students to practice the magic trick with a partner. The cryptology activity was developed from scratch at Colorado School of Mines. The fruit vendor FSA activity is included on the CS

Unplugged website as an alternative to the "Treasure Hunt" activity. We added the assessment worksheets. The minimal spanning tree activity uses the "Muddy City" activity but with an additional worksheet for students to attempt after learning about Kruskal's algorithm. The searching activity replaces the "Battleship" exercise with two similarly structured activities, one dealing with raffle tickets and one with a dragon attacking cows.

In addition to these activities, we have deployed five others: Image Representation (added an assessment worksheet), Artificial Intelligence (added a career extension), Computer Vision (developed from scratch), Routing (developed from scratch) and Sorting (added more whole-class discussion and an assessment worksheet). See [1] for details.

8. CONCLUSIONS

Using CS Unplugged activities for after school groups or other demonstrations have been shown to be sufficiently engaging to keep kids' attention in the lesson. When being deployed in classrooms, however, the activities need more structure and content. Since computational thinking is fairly new, and not understood well by those outside of computer science, the activities need to provide methods to measure and convey what students are learning so that teachers and school administrators can better justify supporting these topics.

In this paper we have presented an approach that supplements CS Unplugged activities with worksheets that relate specifically to CT skills and are designed to capture students' understanding. Our first pilot test with these activities and worksheets helped us identify a number of issues. The second pilot showed improvement, but the results were not as high as we would have liked. After a final round of revisions, we achieved results greater than 50% on all questions, with many results greater than 80% and some approaching 100%.

A few lessons that we learned from our pilot tests:

- A priming activity, in which students attempt to solve a problem in a naïve fashion (e.g., attempting Muddy City before knowing Kruskal's algorithm), is often useful to focus students' attention.
- Students need individual practice to fully grasp a concept. Whole class demonstrations help to engage students and introduce material, but are not sufficient for producing desired results on assessment worksheets.
- Instructions on worksheets must be kept to a minimum and edited carefully so that students can quickly attempt the exercise. Any confusion about what they need to do will result in many students not even attempting the activity. Logistics must be carefully considered to ensure all students have a role to play (any amount of down time potentially results in bored, disengaged students and lower assessment results).
- Vocabulary on worksheets must be consistent with how topics are presented.
- Real world connections help to engage students' interest, but examples must be relevant to the audience (i.e., middle school students).

The results from the spring 2016 deployment are promising. Much remains to be done, however, to a) determine the range of topics that are appropriate for middle school students, b) design additional worksheets that are both engaging for the students and map well to CT concepts, and c) incorporate these worksheets and assessments into lesson plans that can be used by middle school teachers with limited knowledge of CT.

9. ACKNOWLEDGMENTS

The authors would like to thank the CS Unplugged team at Mines (Stephen Kennicutt, Nicholas Dyer, Shelly Konopka, Mykel Allen, Vy Ta and Martin Kuchta) for their assistance in refining, deploying and assessing the CS Unplugged activities. This material is based upon work supported by the National Science Foundation under Grant Numbers CNS-1240964 and DGE-0801692. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

10. REFERENCES

[1] CS Unplugged at Mines. http://toilers.mines.edu/CS-Unplugged. Retrieved 28 April 2016.

[2] Computer Science Teachers Association. Operational definition of computational thinking. http://csta.acm.org, 2011. Retrieved 28 April 2016.

[3] CS Unplugged. About CS Unplugged. http://csunplugged.org, 2015. Retrieved 28 April 2016.

[4] Q. Cutts, S. Esper, and B. Simon. Computing as the 4th "R": A general education approach to computing education. In *International Workshop on Computing Education Research*, Rhode Island, 2011.

[5] Exploring Computer Science. http://exploringcs.org. Retrieved 28 April 2016.

[6] D. D. Garcia and D. Ginat. Demystifying computing with magic. In *Special Interest Group on Computer Science Education*, Raleigh, 2012.

[7] L. Lambert and H. Guiffre. Computer science outreach in an elementary school. *Journal of Computing Sciences in Colleges*, 24(3):118–124, 2009.

[8] T. Nishida, S. Kanemune, Y. Idosaka, M. Namiki, T. Bell, and Y. Kuno. A CS Unplugged design pattern. In *Special Interest Group on Computer Science Education*, Chattanooga, 2009.

[9] R. Taub, M. Ben-Ari, and M. Armoni. The effect of CS Unplugged on middle-school students' view of CS. In *Conference on Innovation and Technology in Computer Science Education*, Paris, 2009.

[10] R. Thies and J. Vahrenhold. Reflections on outreach programs in CS classes: Learning objectives for "unplugged" activities. In *Special Interest Group on Computer Science Education*, Raleigh, 2012.

[11] R. Thies and J. Vahrenhold. On plugging "unplugged" into CS classes. In *Special Interest Group on Computer Science Education*, Denver, 2013.

[12] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.