Assessing Computational Thinking in CS Unplugged Activities

Brandon Rodriguez, Stephen Kennicutt, Cyndi Rader, and Tracy Camp
Division of Computer Science
Colorado School of Mines, Golden, CO, USA
brandonrrodriguez@gmail.com, stephen.kennicutt@gmail.com, crader@mines.edu, tcamp@mines.edu

ABSTRACT

Computer Science (CS) Unplugged activities have been deployed in many informal settings to present computing concepts in an engaging manner. To justify use in the classroom, however, it is critical for activities to have a strong educational component. For the past three years, we have been developing and refining a CS Unplugged curriculum for use in middle school classrooms. In this paper, we describe an assessment that maps questions from a comprehensive project to computational thinking (CT) skills and Bloom's Taxonomy. We present results from two different deployments and discuss limitations and implications of our approach.

CCS Concepts

•**Social and professional topics → Computational thinking;** *Student assessment; K-12 education;*

Keywords

Assessment, Computational Thinking, CS Unplugged

1. INTRODUCTION

Recent initiatives have called for more computer science education at all levels from kindergarten through high school [3]. The International Society for Technology in Education (ISTE) 2016 standards for students include Computational Thinking (CT) as one of the core competencies to thrive in a technological world [2]. Since the term "Computational Thinking" was coined in 2006 by Jeanette Wing [20], numerous researchers have proposed methods to inject CT into the classroom (e.g., [11]). One potential approach is to utilize a set of activities known as CS Unplugged [1, 6].

CS Unplugged activities were developed to encourage interest in computer science. The activities are kinesthetic, engaging, and accessible for students without a computer or typing skills. Studies have shown that CS Unplugged activities can stimulate interest in Computer Science when used in a variety of settings [7]. An analysis based on Bloom's taxonomy, however, showed that the original activities were not suitable as standalone teaching units in middle school classrooms [17]. Similar concerns have been expressed regarding the use of CS Unplugged activities in high school [8]. By embedding the activities into a traditional lesson, which included additional worksheets and discussion, the authors of [17] showed that Unplugged activities could be as effective as alternate methods for teaching fundamental computing concepts such as binary numbers, binary search, and sorting networks [18].

Over the last three years, our research team has modified and enhanced 10 of the CS Unplugged activities for use in middle school classrooms. Since classroom time is extremely valuable, it is imperative for us to illustrate that students participating in the activities are learning computational concepts. In this paper, we present an assessment strategy based on comprehensive projects that incorporate topics from multiple Unplugged activities.

2. RELATED WORK

It took several years and much discussion for the computing education community to reach consensus about what categories of problem solving fit under the umbrella of CT. More recently, researchers have been proposing techniques to measure CT skills. Since CT emphasizes processes over facts, and open-ended solutions rather than exact answers, developing an effective assessment strategy is not trivial.

One common technique to assess CT is to analyze programs created by students. This approach was used by the researchers who developed the Agentsheets visual programming environment [4]. The team analyzed the rules students used in their programs to uncover nine Computational Thinking Patterns, which are constructs students use in programming games that can also be applied to simulations. They have developed an automated tool for Computational Thinking Pattern Analysis that provides visual feedback regarding the extent to which students' artifacts incorporate the desired computational thinking patterns [10].

A related strategy is to have students complete assessment tasks in a specific language. Werner et al. designed an assessment in Alice that consisted of three tasks [19]. They related these tasks to two aspects of CT: thinking algorithmically and making effective use of abstraction and modeling. The tasks were also associated with comprehension (i.e., modifying an existing program), design, and problem solving. The authors noted that it was important for the assessment tasks to use language that was consistent with

SIGCSE '17, March 08-11, 2017, Seattle, WA, USA
© 2017 ACM. ISBN 978-1-4503-4698-6/17/03...$15.00
DOI: http://dx.doi.org/10.1145/3017680.3017779

the game characters. We have also modeled our assessments to have brief "stories" similar to the Unplugged activities.

Some researchers have noted that programs should not be the only tool used to evaluate students' understanding. Grover et al. argue that multiple forms of assessment are needed to effectively evaluate CT [9]. In addition to using rubrics to score students' open-ended projects, this team reused quiz questions from the Israel National Exam to provide an objective measure of students' comprehension of algorithmic flow of control in Scratch.

SRI International and the Educational Testing Service (ETS) have suggested an assessment paradigm based on Evidence-Centered Design, or ECD. ECD assessments answer the questions "What skills should be assessed?" and "What student performances reveal those skills?" ECD arrives at an answer to these questions by first identifying the focal knowledge, skills, and attributes (FKSA) for the domain. Once the CT practices FKSA have been identified, they are mapped to specific curricular units and potential work products that can be scored to provide evidence that students have met the learning objectives [12, 16].

The Bebras International Contest on Informatics and Computer Fluency (Bebras) is a set of computation-related challenges that embody several components of CT [13]. Although not affiliated with CS Unplugged, the Bebras activities can also be used without a computer.

3. COMPREHENSIVE PROJECT ASSESSMENT

Designing the assessments to be engaging for students (i.e., not another standardized multiple choice test) is an important step for keeping students interested in CT, and helps broaden participation. Drawing upon ideas presented in [12, 13, 16], our assessment approach relies on a comprehensive final project that presents a series of problems that are variations of the Unplugged activities. The questions are intentionally open-ended so as not to directly lead students to a desired outcome.

3.1 Project Description

Two versions of our final project were developed for use in our study design. The two projects were designed to be isomorphic, such that a concept tested by one question in the first project would be tested in a similar fashion by one question (a different question) in the second project. In this way, students would not become overly familiar with the wording or context of one version of the problem, and write down a "remembered" solution when completing the second project. Furthermore, each question was designed to be completed independently of all other questions. Question orthogonality allowed students who were unable to complete a problem to advance to another part of the project. Each project had its own story or theme, presented as a challenge for the students to solve. In the first project version, which will be called Pets, the goal was to determine which animal got into the cookie jar. The second project version, referred to as Carney, was carnival themed, and had students use clues from various Unplugged types of problems to solve the murder of a carnival employee.

3.2 Project Questions

The comprehensive project included six questions. A rubric was created for each question. Since the entire project needed to be completed within one 50-minute class period, the questions typically did not ask students to show their work or explain their reasoning. The rubrics therefore included only three levels: *Proficient*, *Partially Proficient*, and *Unsatisfactory*. 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.

The questions are listed below, including a brief description of the corresponding CS Unplugged activity and the requirements for an answer to be scored as *Proficient* or *Partially Proficient* (all other answers would be scored as *Unsatisfactory*).

Character Encoding / Binary Numbers. In the *Parity & Error Detection* lesson, students convert binary numbers to letters and use parity bits to identify errors. For this final project question, students were given a list that mapped uppercase letters to decimal numbers. We wanted to ensure the binary numbers students had to convert were simple, so our encoding scheme mapped A = 1, B = 2, etc. Students were asked to decode a word consisting of six 5-bit numbers. Solving this challenge required students to a) recognize that the five-bit codes were in binary, b) remember how to convert from binary to decimal, and c) understand that this decimal number could be used to look up a letter. Students were rated as *Proficient* if they correctly converted all 5 letters or *Partially Proficient* if they decoded some but not all the letters correctly.

Binary Search. The *Binary Search* lesson begins with a search on an unsorted list, followed by a whole class discussion of binary search and a worksheet to reinforce the concept. The related Pets question described a scenario involving a row of shoes placed in order by shoe size. Students were asked for the maximum number of shoes that Delilah the dog would need to dig up to find the desired shoe. The Carney question was similar but related to books sorted alphabetically. Solving this challenge required students to a) recognize that a binary search could be used and b) perform the necessary divisions to determine how many shoes would need to be viewed (middle school students have not studied logarithms, so continuous dividing in two would be the strategy). Students were rated as *Proficient* if they gave a correct answer or *Partially Proficient* if they were off by 1.

Minimal Spanning Tree (MST). The *Muddy City* Unplugged lesson challenges students to connect all houses in a village using the minimum number of stones. The Pets MST question presented students with a connected, weighted graph and asked students to shade the tunnels that would be needed (i.e., the edges) to minimize the number of feet required to connect all the ant hills (i.e., nodes). The Carney project was similar but used a railroad context, where the goal was to shade the tracks that would be needed to connect all cities where the carnival visited using the fewest miles of track. Solving this challenge required students to a) recognize the type of problem and b) remember/apply an effective algorithm (Kruskal's was covered in the lesson) to solve the problem. Students were rated as *Proficient* if they constructed a correct MST or *Partially Proficient* if they created an MST with only 1 or 2 additional edges.

Finite State Automata (FSA). The Unplugged lesson that covers *FSA* first has students discover the rules that govern interactions between a fruit vendor and customer, followed by a lecture that introduces FSA and two worksheets to reinforce interpreting and creating FSAs. The Pets FSA problem provided students with sentences that described the set of rules Delilah the Dog followed to fill her day. States were highlighted in bold and events were underlined. The first FSA question asked students to organize the events to see the possible schedules (FSA Creation). To avoid biasing the results, the wording of the question did not mention FSAs. The second question (FSA Usage) required students to review three possible schedules and identify which one(s) were correct. The Carney questions were similar, but students were given a set of rules for a robotic fortune teller. Solving this problem required students to a) recognize that FSA would be an effective representation, b) remember the "syntax" for creating an FSA, and c) understand how the FSA can be used to identify valid sequences of events. Students rated as *Proficient* were able to create a valid FSA (part 1) and identify two valid sequences (part 2). A *Partially Proficient* rating was assigned for part 1 if students made a recognizable attempt at an FSA but had errors (e.g., extra or missing transitions) and for part 2 if students identified one of the two valid sequences.

Binary Numbers. The *Binary Numbers* lesson uses flip cards to teach students how to convert between binary and decimal. Since the character encoding question encapsulated both binary number conversions and character encoding, an additional question was used to test just binary numbers. This question listed locations that were labelled with a binary number, e.g., Couch = 01111, and characters that were labelled with a decimal number, e.g., Larry 10, Buzzy 15, etc. Students were required to a) convert each binary number to decimal and b) match the decimal number to the correct animal/person. Students were rated as *Proficient* if they correctly identified all 6 locations and *Partially Proficient* if they identified at least 3.

3.3 Project Question Refinements

The pilot version of these projects, as described in [14], consisted of five questions. The pilot test results indicated that the general approach of using a comprehensive project had promise, but highlighted a number of issues with the specific questions, which we fixed for the following deployments and present here as lessons learned.

Extra hints. To avoid frustrating the students who had not seen any CS Unplugged activities, the binary encoding question for the pilot pretest contained explicit hints that were not included in the posttest. The results therefore showed a non-significant decline from pretest to posttest.

Confusing instructions. Several questions were revised due to confusing format or instructions: For example, the MST problem had a hint to start with a specific node, which caused some confusion as a few students interpreted this as a traveling salesman problem. The wording was also modified to emphasize that the goal was to minimize the number of feet (of tunnels or tracks), not necessarily the number of edges.

Consistent format. The FSA challenge highlighted states, but did not specify transitions/events in a clear fashion. The questions were changed to match the format of the worksheets used during the lesson.

Increased complexity. The decoded words in the posttest version of the Character Encoding problem were represented as horizontal and vertical lines, rather than 0s and 1s (e.g., "||-|" rather than "1101"). Thus, the Carney version included an additional level of abstraction not present in the Pets version.

Unrelated question. The pilot included an optimization problem that was inspired by a question from the Bebras challenge. This question did not relate specifically to any of the Unplugged activities and was therefore removed after the initial pilot.

Lesson revisions. In addition to issues with specific questions, analyzing the results also identified concerns with several of the Unplugged activities. For example, the FSA activity jumped too quickly into the "create" mode, which was the skill tested on the comprehensive project. The lesson plan was revised to walk students through the progression of understanding the FSA notation, using an existing FSA, then creating their own FSA based on a set of rules. Similar revisions were made to a number of the lesson plans.

3.4 Bloom's Taxonomy and Computational Thinking

Thies and Vahrenhold [17] noted that the original CS Unplugged activities tended to fall in the lower spectrum of Bloom's taxonomy. To be suitable for middle school classrooms, they recommended that additional depth be incorporated into the activities. This same critique can be applied to assessments. We therefore mapped Bloom's levels of thinking to our projects. This update was done in conjunction with two teachers from our deployment school. Teachers individually evaluated the problems before discussing and justifying their choices. The projects were found to demonstrate all levels of Bloom's Taxonomy (creating, evaluating, analyzing, applying, understanding, and remembering), which make them better poised to measure middle school learning than if they were unilaterally located in the bottom tiers.

After classifying the problems per Bloom's Taxonomy, CT pillars were mapped to each problem. CT encompasses the concepts of data representation, decomposition, pattern recognition, abstraction, and algorithmic thinking. Five members of our research group independently categorized the project problems before aggregating the results. All five members were familiar with the principles of CT as well as the CS Unplugged activities used in the projects. Dr. Tim Bell, the creator of the original CS Unplugged activities, also provided feedback on the project assessment, including mapping each question to its CT skills [5]. As shown in Table 1, the project assessments were judged to exhibit three of the five CT skills. Of the remaining skills, pattern recognition was assessed via worksheets the students completed during the binary numbers and cryptology activities [15]. Problem decomposition was not a focus of any of our assessments.

4. DESIGN STUDY

The results presented in this paper are from two deployments at two different times. Both deployments were done in 7^{th}-grade classrooms and included the comprehensive project plus six CS Unplugged activities. The activities were chosen based on what we believe to be the most appropriate content for a 7^{th}-grade audience. The first deployment, from fall 2015, included a full study design where classes were sep-

Table 1: Each topic of the comprehensive project along with its associated "story" for the Pets and Carney final versions, the CT skills, and the categories for Bloom's taxonomy.

Topic	Pets Story	Carney Story	CT Skills	Bloom's
Character encoding	Melanie Mouse secret message	Detective secret message	Data representation	Remembering Understanding
Search	Delilah Dog shoe search	Odin book search	Algorithmic Thinking	Evaluating
MST	Tunneling ants	Railroad	Abstraction Algorithmic Thinking	Applying
FSA	Delilah Dog schedule	Fortune telling robot	Abstraction	Creating Analyzing
Binary Numbers	Animals hiding	Carney workers hiding	Data representation	Remembering Understanding

arated into a *retention* group and a *knowledge progression* group (pretest/posttest comparison). One graduate student researcher taught all lessons. The second deployment, from spring 2016, assessed results only based on *knowledge progression*. Lessons were taught by a mix of graduate and undergraduate student researchers. Although the spring 2016 deployment results are generally better, due to improvements in the activities and/or assessment questions, we include a few results from the fall 2015 pilot primarily to highlight retention.

Group 1 completed both versions of the project *after* the students learned the CS concepts (see Figure 1). The students completed their normal classroom assignments, not related to any of the CS Unplugged activities, during the six-day period between completing the first and second projects. Results from this group are presented below as the *Retention* study (see Section 5.1).

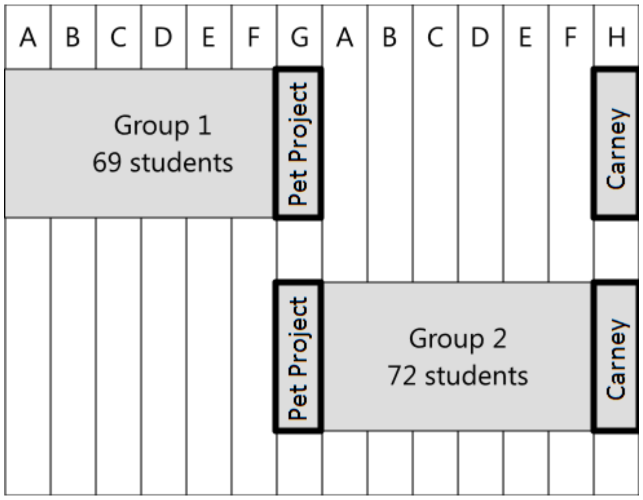

Figure 1: The fall 2015 deployment schedule. Each column is one school day, and each letter in a column represents a unique activity being deployed. Bolded columns signify dates when students completed a comprehensive project.

Group 2 completed one project before seeing any of the CS Unplugged activities, and completed the second project directly after seeing the activities. The spring 2016 deployment matches group 2, as students completed the Pets project as a pretest, followed immediately by the six activities and the Carney project as a posttest. Results from the spring deployment, which included 121 students, are presented below as the *Knowledge Progression* study (see Section 5.2).

5. RESULTS

Using the rubrics described in Section 3, every problem was individually scored as *Proficient*, *Partially Proficient* or *Unsatisfactory* by two researchers. Disagreements on any score were resolved by having both researchers score the question together, and then editing the rubrics to better document any edge cases.

5.1 Retention Study

In this section we address the question "Do students retain information from the CS Unplugged activities after a delay?" Figure 2 shows the change in proficiency in Group 1 between the Pet project (posttest for Group 1) and the Carney project (retention test). As expected, the number of students attaining proficient scores was generally lower on the retention test. According to a χ^2 test, however, none of the differences are significant. Although six days is clearly not long enough to measure long-term retention, we are encouraged that many students remembered at least some of the concepts they learned via the Unplugged lessons (i.e., it is not all just fun).

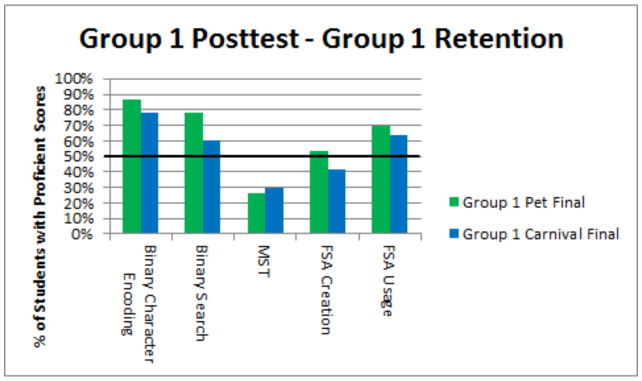

Figure 2: Chart of proficient scores in the posttest and retention test. Scores from fall 2015 deployment.

5.2 Knowledge Progression Study

In this section, we answer the question "Are students learning information from the CS Unplugged activities?" The results presented here are based on the final version of the comprehensive projects.

Figure 3 shows the percentage of students who achieved proficiency in each problem of our final projects. A critical first step in educational research is to ensure that assessments measure the effect of an intervention and not prior knowledge. Table 2 shows that all six problems show significant increases from pretest to posttest, based on the χ^2 test.

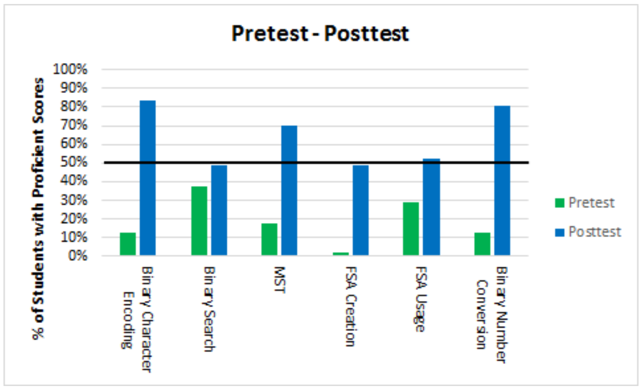

Figure 3: Chart of proficient scores in the pretest and posttest test. Scores from spring 2016 deployment.

In all of the project problems, approximately 50% (or higher) of students who had seen the activities reached proficiency in the posttest. Student performance was highest on the two questions that involved binary number conversion, with approximately 80% of students achieving the *Proficient* level. Most topics in our study are covered in just one 50-minute session. The one exception is binary numbers, which are reinforced when students learn about ASCII encoding schemes. This second exposure may help to explain why a higher percentage of students reached the *Proficient* level for these questions.

Most students (70%) were also able to successfully identify the edges that were needed to create a minimal spanning tree. This result is encouraging, because the pretest results show that, without knowing a strategy, most students could not successfully complete the task. But, after being introduced to a fairly sophisticated algorithm (Kruskal's), students were able to recognize a similar problem and solve it successfully. Note that we have no way to know whether students identified as *Proficient* actually followed Kruskal's algorithm. In fact, we believe that memorizing the details of one algorithm is not as important as being able to systematically approach a problem of the same type. It seems that students who reached the level of *Proficient* on this problem have made progress toward that goal.

We are also encouraged by the fact that close to 50% of the students were able to generate an FSA drawing based on a set of rules. As shown in Figure 4, an additional 17% were ranked as *Partially Proficient*, meaning that they used some of the boldface events and states in a diagram, but did not create a complete FSA.

The number of *Proficient* scores on the binary search problem is only around 50%. As part of the binary search lesson, students are asked to perform a binary search on a set of ordered items. The comprehensive project question

Table 2: Results of a χ^2 test for each lesson deployed in spring 2016. All results are significant.

Activity	**χ^2 Value**
MST	2.27e-16
FSA Construction	3.63e-20
FSA Validation	1.29e-4
Binary Character Encoding	2.44e-27
Binary Search	1.17e-3
Binary Number Conversion	1.4e-25

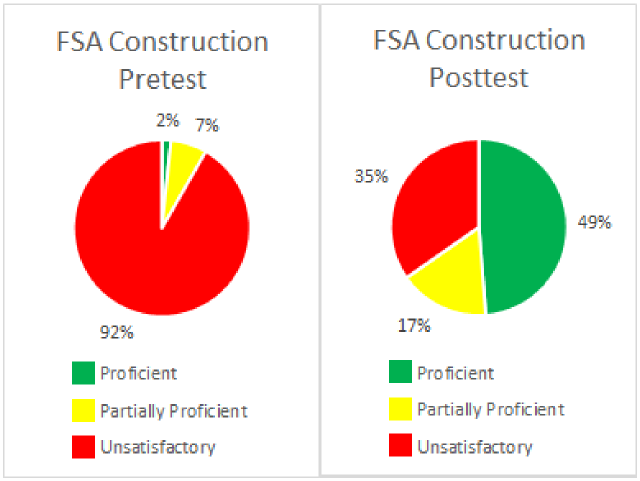

Figure 4: Results from our deployment of the FSA lesson plan in spring 2016.

is more abstract because students are not explicitly asked to perform a binary search. Instead, students must recognize that a binary search could be used, then think about how to determine the number of comparisons. Although counting the number of comparisons was illustrated during the lesson, students completing the worksheets were never explicitly asked to do that calculation. It is possible that this question is not developmentally appropriate (i.e., too challenging) for middle school students.

In analyzing the incorrect answers provided for this question, 24 students (19.8%) gave an answer of 15 or 16 comparisons (for a list of 16 items). These students evidently thought that it would be necessary to look at every item to find the desired one. Although they missed the idea of binary search, the students appear to be thinking about the requirements of the problem. Ten students (8.3%) listed the answer as 8. These students may have remembered at least some ideas from the lesson (i.e., halving) but did not recall the entire process. For the remaining answers (23.1%), it was not clear what the students were thinking. These results are shown in Table 3.

6. DISCUSSION

When using activities designed to promote enthusiasm for computing, such as CS Unplugged, it's important to ask a) whether students are actually learning the intended concepts and b) whether they will remember anything from the lesson other than whether they did (or did not) enjoy the activity. In this paper, we have attempted to answer these questions

Table 3: Scoring breakdown of the binary search problem from the spring 2016 deployment.

# Comparisons	# Students
4 or 5 (correct)	59
15 or 16 (linear search)	24
8 (cut in half once)	10
3 or 6 (potentially off by 1)	15
0, 2, 11, 12, 13, 14, 26	13

via comprehensive projects that relate specifically to five Unplugged activities (the sixth activity did not have an associated question and is, therefore, not discussed here). CT skills are fairly broad, so it is clearly not possible to claim that students have mastered a particular CT component based on just one assessment. Our approach uses assessment questions that are similar in style to worksheets performed as part of the lesson. We make no claims, for example, that performing well on the binary character encoding, which we have mapped to the CT skill of data representation, will result in students understanding a completely different data representation task (e.g., images).

Furthermore, it can be difficult to map proven computer science techniques to specific CT areas. For example, our team assigned creation and validation of FSA to the CT skill of abstraction, but it could be argued that creating a FSA should be categorized as algorithmic thinking. With the increased emphasis on K-12 computing, it is critical that the community develop assessments so that teachers, parents, and administrators can understand what students are learning. CT skills provide a valid framework for this effort, but forcing every assessment of computing skills to fit neatly within CT may be counterproductive.

Lesson plans for all Unplugged activities described in this paper are available on our website [1]. The authors are happy to share final versions of the Pets and Carney projects upon request.

7. ACKNOWLEDGMENTS

The authors would like to thank the CS Unplugged team at Mines (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.

8. REFERENCES

[1] CS Unplugged at Mines. http://toilers.mines.edu/CS-Unplugged. Retrieved 11 December 2016.

[2] ISTE standards for students. http://www.iste.org. Retrieved 11 December 2016.

[3] Computer science for all, 2016. http://whitehouse.gov. Retrieved 11 December 2016.

[4] A. R. Basawapatna, K. H. Koh, and A. Repenning. Using scalable game design to teach computer science from middle school to graduate school. In *Conference on Innovation and Technology in Computer Science Education*, Bilkent, 2010.

[5] T. Bell. Personal Communication, October 2015.

[6] CS Unplugged. About CS Unplugged. http://csunplugged.org. Retrieved 11 December 2016.

[7] P. Curzon, Q. Cutts, and T. Bell. Enthusing & inspiring with reusable kinaesthetic activities. In *Proc. 14th Conference on Innovation and Technology in Computer Science Education*, Paris, France, July 2009.

[8] Y. Feaster, L. Segars, S. Wahba, and J. Hallstrom. Teaching CS Unplugged in the high school (with limited success). In *Proc. 16th Conference on Innovation and Technology in Computer Science Education*, Darmstadt, Germany, June 2011.

[9] S. Grover, S. Cooper, and R. Pea. Assessing computational learning in K-12. In *Conference on Innovation & Technology in Computer Science Education*, Uppsala, 2014.

[10] K. Koh, H. Nickerson, A. Basawapatna, and A. Repenning. Early validation of computational thinking pattern analysis. In *Proc. 19th Conference on Innovation and Technology in Computer Science Education*, Uppsala, Sweden, June 2014.

[11] I. Lee and K. Apone. Integrating computational thinking across the K-8 curriculum. *ACM Inroads*, 5(4):64–71, December 2014.

[12] R. J. Mislevy, R. G. Almond, and J. F. Lukas. A brief introducation to evidence-centered design. Technical report, Princeton University, 2003.

[13] W. Pohl and V. Dagiene. Bebras international contest on informatics and computer fluency. http://bebras.org. Retrieved 11 December 2016.

[14] B. Rodriguez. Assessing computational thinking in Computer Science Unplugged activities. Master's thesis, Colorado School of Mines, Golden, 2015.

[15] B. Rodriguez, C. Rader, and T. Camp. Using student performance to assess CS Unplugged activities in a classroom environment. In *Innovation and Technology in Computer Science Education*, Arequipa, July 2016.

[16] D. W. Rutstein, E. Snow, and M. Bienkowski. Computational thinking practices: Analyzing and modeling a critical domain in computer science education. In *American Educational Research Association*, Philadelphia, 2014.

[17] 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.

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

[19] L. Werner, J. Denner, and S. Campe. The fairy performance assessment: Measuring computational thinking in middle school. In *Proc. 43rd ACM Technical Symposium on Computing Science Education*, Raleigh, USA, February 2012.

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