Empowering All Students: Closing the CS Confidence Gap with an In-School Initiative for Middle School Students

Philip Sheridan Buffum
psbuffum@ncsu.edu[1]

Megan Hardy Frankosky
rmhardy@ncsu.edu[2]

Kristy Elizabeth Boyer
keboyer@ufl.edu[3]

Eric N. Wiebe
wiebe@ncsu.edu[4]

Bradford W. Mott
bwmott@ncsu.edu[1]

James C. Lester
lester@ncsu.edu[1]

[1] Computer Science, North Carolina State University, Raleigh, North Carolina, USA
[2] Psychology, North Carolina State University, Raleigh, North Carolina, USA
[3] Computer & Information Sciences & Engineering, University of Florida, Gainesville, Florida, USA
[4] STEM Education, North Carolina State University, Raleigh, North Carolina, USA

ABSTRACT
The important goal of broadening participation in computing has inspired many successful outreach initiatives. Yet many of these initiatives, such as out-of-school activities or innovative new computer science courses for secondary school students, may disproportionately attract students who already have prior interest and experience in computing. How, then, do we engage the silent majority of students who do not self-select computer science? This paper examines this question in the context of ENGAGE, an in-school outreach initiative for middle school students. ENGAGE's learning activities center on a game-based learning environment for computer science. Results reveal that the initiative improved the computer science attitudes of students who were not already predisposed to study computer science, in a way that a corresponding after-school program could not. The results illustrate how an in-school initiative can empower young students who might not otherwise consider studying computer science.

Categories and Subject Descriptors
K.3.2 [**Computers & Education**]: Computer and Information Sciences Education --- *Computer Science Education*

Keywords
In-school outreach, middle school, broadening participation.

1. INTRODUCTION
Computing skills are increasingly integral to many 21st century jobs. Nonetheless, computer science remains a "niche" subject in the United States, studied by a generally small, non-diverse population of students. Consequently, many current initiatives in the United States seek to broaden participation in computing. Many of these initiatives involve out-of-school activities such as summer camps [1]. Other initiatives focus on formal, in-school coursework through the development of innovative pre-college curricula such as Exploring Computer Science [10] and AP Computer Science Principles [2, 11]. Fundamental to all of these initiatives is the mission to engage students who are historically underrepresented in computer science, including female students.

Yet for both of these types of initiatives—out-of-school activities and in-school computer science courses for secondary school students—questions remain as to just how effective they are in broadening participation [17]. Both these types of initiatives are likely to involve self-selection, attracting a certain subset of students rather than a truly representative sample. In other words, even if a given initiative has a large percentage of underrepresented students (e.g., female students) among its participants, these students likely either already have interest in studying computer science or have someone in their lives encouraging them to study computer science. Of course, this does not negate the value of these initiatives: nurturing a student's pre-existing interest in computing and empowering her with new skills may contribute greatly to her persisting in the field.

To fully address the goal of broadening participation, however, we must also reach out to students who do not have predispositions or existing influences to study computer science or to participate in extracurricular computing activities. This paper provides evidence that the key is to develop *in-school* initiatives that enroll a broad population of students (i.e., not students specifically seeking a computer science elective). Furthermore, it is crucial to create such in-school initiatives at the *pre-secondary level*, as students begin their career trajectory as early as middle school [14]. Indeed, researchers have looked specifically at how the underproduction and underrepresentation issues in undergraduate computer science departments may be traced back to lack of exposure as early as middle school [25, 28].

Our team has embarked on one such initiative, in which we hope to leverage the engaging nature of game-based learning environments to spark interest in computing. Over the past three years, we have developed ENGAGE, an immersive game-based learning environment that adapts learning objectives from the AP CS Principles course [2] to be middle-grade appropriate. We then integrated this computing content with an existing middle school science curriculum and trained middle school science teachers to teach this course in four diverse middle schools. The results show that the in-school implementation of ENGAGE improved the computer science attitudes of students with no prior computer science experience, in a way that the corresponding after-school implementation could not.

 SIGCSE '16, March 02-05, 2016, Memphis, TN, USA © 2016 ACM. ISBN 978-1-4503-3685-7/16/03 $15.00
DOI: http://dx.doi.org/10.1145/2839509.2844595

This paper explores the potential of situating an outreach initiative within an in-school course for middle school students. Section 2 provides a review of related work, highlighting three elements that can make in-school initiatives successful. Section 3 describes the development and in-school implementation of ENGAGE. This includes software development, curriculum development, and teacher development. Finally, Section 4 reports on the initiative's success at improving computer science attitudes of those students who were less predisposed to study computer science. Overall, the paper highlights the potential of a pre-secondary, in-school initiative to reach students who might not otherwise consider computer science as a field of study, and to improve their attitudes toward computer science.

2. RELATED WORK

To attain a full understanding of the recent work on in-school computer science outreach initiatives, we performed a comprehensive literature review of the past five SIGCSE conferences (2011-2015). Of the approximately 500 full papers presented at these conferences, we identified 113 papers focusing on K-12 education. Slightly more than half of these K-12 papers reported on out-of-school initiatives (e.g., summer camps), and another quarter of them focused on in-school computer science electives for secondary school students, such as AP Computer Science Principles. This paper reviews the remaining literature (in-school, pre-secondary) that address the key question: *How can we create in-school outreach initiatives that reach those students who do not necessarily have a predisposition to study computer science?* Performing studies on such a general population of students, rather than on a subset of students who self-select to attend computer science activities, can provide greater insight into how all students learn computer science, regardless of prior knowledge or interest. For example, in-school research studies have examined the programming abilities of primary students as related to their reading abilities [24] and to their general academic performance in other subjects [19], in both cases providing better understanding on how *all* students learn programming, and not just those who self-select computer science. Crucially, research has also looked at how to design in-school initiatives with an emphasis on sustainability [13].

Reaching K-12 students with computer science content during school hours ultimately requires empowering K-12 teachers to teach that computer science content. It is important to note that the instructors for after-school and summer initiatives are often different than those who would teach an in-school course for middle school. Accordingly, a significant amount of research in this area has addressed *teacher professional development*. Perhaps the most popular approach for this has been to introduce K-12 teachers to computer science content through summer workshops [12, 16]. Other research has looked at introducing computer science content to student teachers through pre-service teacher education [3, 30], or to in-service teachers with research experience in computer science labs at local universities [26]. While these workshops have been shown to be effective at having an immediate impact on teachers' knowledge and attitudes, less research has been done to examine how likely teachers are to keep teaching the content year after year [13]. Moreover, little research has been reported on how effectively these teacher benefits translate into measurable impacts on *students*, and particularly on students who are not predisposed to study computer science.

We might reasonably expect greater and more sustainable impacts if teachers have an established *curriculum* to accompany their professional development. One middle school curriculum uses a "braided teaching" approach by interleaving a number of computer science concepts throughout a course for middle school students [20]. Another middle school curriculum focuses on improving computer science attitudes by highlighting the breadth of the field, an approach that should perhaps start at the primary school level [8]. Several recent efforts have created pre-secondary computer science curricula by leveraging existing tools also popular with out-of-school initiatives, such as Scratch [23] and Alice [21]. These pre-existing tools offer great functionality. The sheer breadth of possibilities that an open-ended learning environment like Scratch affords, however, can make it difficult for K-12 teachers unfamiliar with computer science to ensure that student work is adhering to a curriculum's learning objectives, perhaps making it less likely that the teachers will fully adopt it [15, 18]. More work needs to be done to investigate whether relying on these ubiquitous tools privilege students who have already used them in prior computer science activities.

Most initiatives involve some software system, but there has been comparatively little recent work that reports on developing software systems specifically for in-school initiatives. The research that has been reported has led to valuable insights. AgentSheets has been used to assess middle school students' computational thinking [4]. Most importantly for the goal of broadening participation in computing, research on the AgentSheets system has explored ways to use computer science pedagogy to improve computer science attitudes among female students [27]. Another large-scale effort, Bootstrap, uses its custom software tool WeScheme to integrate computing concepts with middle school algebra [31]. This custom software enables Bootstrap to reinforce algebra content in ways that a pre-existing tool like Scratch would not, and this ability makes it particularly appealing to math teachers [22]. With both AgentSheets and Bootstrap, we see the great potential value of developing software systems tailored to the curriculum.

In the related literature on in-school pre-secondary initiatives, we see the importance of three key types of development: professional development for teachers, development of curricula, and development of software that links strongly to the curriculum. This is not to say that all successful in-school initiatives need each of these elements. Rather, we argue that they each have great potential to help produce in-school initiatives that are sustainable and have positive impacts on a broad population of students. Below we describe how we endeavored to incorporate all three of these elements into a middle school initiative. As part of a three-year project, we first developed a game-based learning environment along with a computer science curriculum, and then ran a Teacher Institute so that we could deploy the resulting course in four diverse schools during the 2014-15 school year. As noted above, our overarching goal at each stage of development was to create an initiative that would reach students who might not otherwise consider computer science, and positively impact their computer science attitudes.

3. DESIGN OF IN-SCHOOL INITIATIVE

3.1 Development Process

Our research and development team developed both the ENGAGE game-based learning environment and its accompanying computer science curriculum synchronously over two years. Although the game-based learning environment[1] can be deployed within a

[1] This game-based learning environment is one in which students learn by playing the game, in contrast to learning by building games.

wider curriculum (including non-gameplay lessons with complementary learning activities), we designed the game so that it could feasibly operate in a stand-alone fashion. Students with no prior computer science knowledge can play the game from beginning to end, developing their computational thinking skills with only the in-game activities scaffolding their learning. This provides more flexibility in how ENGAGE can be deployed, allowing for out-of-school and in-school implementations.

Crucially, designing the game-based learning environment in this way also eases the burden on the teacher for in-school implementations. We anticipated that our middle school teachers would have limited prior computer science knowledge, so we sought to create a software environment that would ease the burden of teaching an unfamiliar subject. We hoped this step would positively influence how likely the teachers would be to teach the course again and maintain "curriculum integrity" [13].

In developing the curriculum for in-school implementation, we planned for a 20-session course, with each session lasting an hour. We envisioned a schedule in which gameplay sessions would alternate with non-gameplay sessions. During a given gameplay session, we planned for students to receive an introduction to a certain computational concepts within the game-based learning environment. Then the next day's session would center on a classroom activity that would reinforce the concept. As we progressed with the development of the game, we thus generated ideas for activities that would extend students' understanding of the material, while not being integral to success in the game.

During game development, we delayed drafting lesson plans for the non-gameplay activities because we wanted to include the eventual classroom teachers as co-creators. Ultimately, with the support of district-level stakeholders, we integrated our computer science content with an existing quarterly science elective focused on oceanography. Our partner schools offer this course to students each quarter of the academic year during normal school hours, and it draws a diverse population of students. The entire course consists of 45 hour-long sessions. Our in-school implementation plan thus became to integrate three types of sessions: gameplay sessions, non-gameplay sessions that reinforced computational concepts, and non-gameplay sessions that focus on the pre-existing science content. We approached this task in close collaboration with classroom teachers, as detailed in Section 3.2.

3.2 Overview of System and Curriculum

The ENGAGE game-based learning environment engages the student in a narrative in which she is a computer scientist tasked with solving a socially relevant mystery. As the student advances through the immersive, three-dimensional game world (Figure 1), she must employ computational thinking skills to overcome challenges. With the curriculum's learning objectives derived from the AP CS Principles curriculum framework, some of these challenges require the student to write programs in a block-based programming interface, but the emphasis is on students deepening their conceptual understanding rather than only developing programs. Additionally, the narrative of the game reinforces the idea of the student becoming a computer scientist. By focusing the game's goals on student conceptual understanding and identity formation, the goal is to avoid overly privileging students with prior programming experiences. The project keenly hopes to empower all students to view computer science as a possible future subject of study.

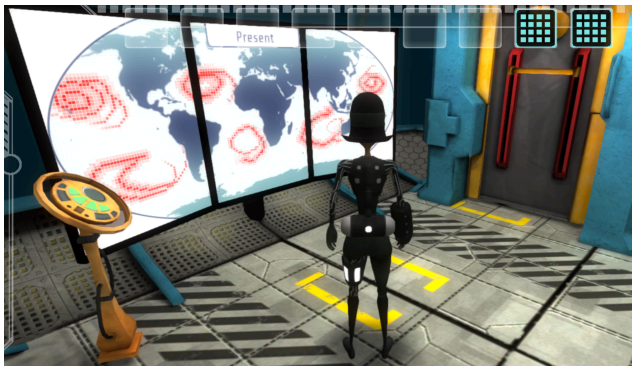

Figure 1. The ENGAGE game-based learning environment.

The game is divided into three levels. The first level introduces students to basic programming concepts such as sequencing and iteration. The students will later apply these concepts in more sophisticated programs in the subsequent levels. The second level focuses on binary numbers. Students learn how a binary system works, how to interpret binary numbers, and how computers use binary numbers to represent other forms of data, such as text and images. The third and final level focuses on exploration of data. Students write programs to make sense of data they receive from other characters in the game. In addition to learning about algorithms such as sorting and filtering data, students solve challenges that require the use of large data sets to solve globally-relevant problems.

3.3 Teacher Institute

In summer 2014, we held a six-day Teacher Institute, which included a total of 40 hours of workshop activities with middle school science teachers. We recruited four teachers from four different middle schools, each serving a diverse student population. In addition to attending the Teacher Institute, these four science teachers all committed to teaching our integrated curriculum during one of their class periods for the 2014-15 school year. As described below, the teachers co-designed this integrated curriculum with the research team, supplementing the research team's *computer science education* and *game-based learning* expertise with the teachers' *science education* and *middle school teaching* expertise.

In preparation for the Teacher Institute, the research team created a teacher kit that included a project overview, a proposed classroom implementation plan, guidelines for survey administration, and a summary of key locations in the game. These key locations consisted of 1) challenges that students might find particularly difficult, and 2) areas in which the teacher might especially impact deeper student learning by asking targeted questions about the given topic. Additionally, the teacher kit contained placeholders for lesson plans that would be created collaboratively during the workshop sessions. We aimed to produce detailed lesson plans specifying the learning objectives, necessary materials, and connection to the content in the game-based learning environment. We also planned for the Teacher Institute to produce an updated in-school classroom implementation plan, clarified by teacher input.

In designing the structure for the Teacher Institute of 2014, we built upon our experience conducting a series of workshops the previous summer [6]. We devoted the first day of this new Teacher Institute to an overview of the project. This included an introduction to the game-based learning environment, a summary of the computer science content, and our vision for how we might

integrate the computer science content with the existing science curriculum. On each subsequent day, we used the morning hours to provide the teachers a solid understanding of both the project and the content knowledge. Since these teachers did not enter with backgrounds in computer science, we needed first to teach them the computer science topics that the students would be learning in the curriculum. We then used afternoon sessions to give the teachers time to draft lesson plans that would incorporate this computer science content into the existing science curriculum.

By the end of the Teacher Institute, we had produced a detailed in-school implementation plan, endorsed by all teachers. As noted above, the entire curriculum was designed for an academic quarter lasting approximately two months. We expected that the implementation plan would remain consistent from quarter to quarter, but, anticipating the need to refine the plan iteratively, we only solidified a calendar for the first academic quarter. The goal was to set dates for certain lessons so that members of the research team could coordinate with the teachers in preparing those class sessions. As described below, members of the research team worked closely with the teachers during the first quarter to troubleshoot issues with the game-based learning environment, to provide content knowledge support to the teachers, and to gather observations on how to improve the classroom implementation.

3.4 Study Implementation

Due to school constraints and other factors, not all teachers taught this curriculum each quarter of the 2014-15 academic year. During the first quarter in which each teacher taught the course, members of the research team attended most gameplay sessions and all non-gameplay sessions that involved computer science content. Teachers had the responsibility of delivering all lessons, but could request assistance from the research team member in the case of any uncertainty. For the most part, the teachers did not need to rely on in-class support from the research team. They reported satisfaction with the training that they received during the Teacher Institute, as well as appreciation for how the game-based learning environment supported instruction.

The classroom study saw over 200 students complete the course during the 2014-15 academic year. Students overwhelmingly responded positively to the experience on post-surveys, and classroom observations further established the game's success at engaging students. Furthermore, preliminary studies have revealed significant learning gains, as measured by ENGAGE's purpose-built knowledge assessment instrument [5, 7]. The rest of this paper, however, will focus on another dimension of the initiative's success: reaching a diverse student population, as measured by computer science attitudes.

4. COMPARISON OF STUDENTS WITH AND WITHOUT PRIOR PROGRAMMING EXPERIENCE

As discussed above, the desire to reach a diverse student population motivated our in-school implementation strategy. The fundamental goal of our project is to broaden participation in computing. While there exist many useful approaches to supporting this grand goal, we focus on reaching *students who might not otherwise consider computing fields*, helping them build their computational thinking skills, and positively impacting the attitudes that they hold towards computer science.

This requires us to go beyond merely looking at traditional demographic information to evaluate our success. In previous pilot studies, conducted as out-of-school initiatives, we succeeded in recruiting a large percentage of female students [6]. Yet the self-selection that is inherent to such out-of-school initiatives allows for the possibility that we may only have been reaching the female students who already felt predisposed to study computer science, or who had parents who encouraged them to study computer science. Furthermore, even if we "hid" computer science content within some out-of-school activity advertised as another discipline, which can be successful at reaching students who are not predisposed to computer science [9], we still might face issues of access that in-school activities largely avoid.

Thus, while we want to ensure that our initiative appeals to student groups generally underrepresented in computer science (and we have, in fact, analyzed ways in which it does so [7]), we also want to measure the success of our initiative at specifically impacting students with less of a predisposition toward studying computer science. To begin examining this, we can compare the computer science attitudes of students based on their prior computing experiences. This paper reports on the data of one our partner schools, Ada Middle School,[2] which offered the course each quarter of the 2014-15 academic year. Ada is an urban middle school serving a racially and ethnically diverse community, with over one third of its students receiving free or reduced lunch. It must be noted that Ada does provide its students a relatively high degree of exposure to computing due to its STEM theme and its proximity to a technology hub.

Of the 84 total students who completed all surveys (including pre- and post-surveys on computer science attitudes), 31 were female students and 53 were male students. As for race or ethnicity, the in-school implementation included 12 African-American students, 19 Asian students, 6 Latino/a Students, 1 Middle Eastern student, 6 mixed/multiracial students, 19 South-Asian students, and 21 white students. Meanwhile, on the survey item, "Have you ever participated in any activities that involve computer science or computer programming?", 48 of the participants reported "yes" and 36 reported "no".

To compare the computer science attitudes of those with and without prior programming experience, we utilized an attitude survey that was originally validated for college students [29], and which was modified for middle school students. This computer science attitudes (CSA) survey includes three subscales: *confidence* in computer science skills, *usefulness* of computer science, and *motivation* to study computer science. Perhaps unsurprisingly, the students with prior programming experience scored higher on this CSA survey overall and on all three subscales, as seen in Table 1.

Table 1. Computer science attitudes (CSA) on pre-survey of students with and without prior programming experience

	Prior Programming	No Prior Programming
CSA Overall	3.64 (SD = .62)	3.11 (SD = .65)
Confidence Subscale	3.88 (SD = .74)	3.08 (SD = .85)
Usefulness Subscale	3.59 (SD = .81)	3.23 (SD = .80)
Motivation Subscale	3.44 (SD = .75)	3.01 (SD = .64)

[2] Names of schools are pseudonyms. We report only on the data from this school here so that we can make a clearer comparison to students in an after-school initiative, described below.

One-way ANOVAs found all these differences to be significant with *p*–values less than .05. The confidence subscale saw an especially stark difference ($p < .001$). We expected to see such differences in computer science attitudes on this pre-survey. Students with prior programming experience may express more positive computer science attitudes because of prior programming experiences they have had. They may also have participated in those prior programming experiences due to their existing predispositions toward studying computer science.

The reinforcing interplay of *participation* and *interest* in computer science activities seems to start early. With this in mind, our in-school implementation aims to have particularly strong positive impacts on middle school students who *do not* have prior exposure to computer programming. To measure the initiative's success in this goal, we can look at the results of the CSA post-survey. As Table 2 shows, the pre-existing differences in computer science attitudes largely disappeared by the time students took this post-survey.

Table 2. Computer science attitudes (CSA) on post-survey of students with and without prior programming experience

	Prior Programming	No Prior Programming
CSA Overall	3.53 (SD = .9)	3.32 (SD = .86)
Confidence Subscale	3.6 (SD = 1.05)	3.41 (SD = .95)
Usefulness Subscale	3.54 (SD = .96)	3.27 (SD = 1.02)
Motivation Subscale	3.46 (SD = .94)	3.29 (SD = .82)

On the post-survey, neither the CSA overall nor any of the subscales revealed any significant differences between students with and without prior programming experience. The most dramatic shift occurred in the confidence subscale, where students with no prior programming experience increased their confidence to near the level of those with prior programming experience. A repeated-measures ANOVA found this increase from pre-survey to post-survey to be statistically significant ($F(1, 35) = 5.039$, $p < .05$). In many ways, this should be expected; following their participation in this initiative, all students on the post-survey now have some prior programming experience, so in some ways they all leave the initiative in the same category. If we had not situated this initiative in-school, however, we may never have reached those students with no prior programming experience. Now, students in this crucial demographic leave the initiative feeling empowered, and perhaps more motivated to study a subject that they had never before considered.

To further illustrate the potential value of in-school initiatives, we compare the in-school implementation to an after-school activity our research team conducted in Spring 2015. For this activity, we advertised for student participants at Hopper Middle School, which is *not* one of the middle schools where our course is offered during the school day, but is in the same district. Compared to Ada, Hopper Middle School has even larger percentages of students from underrepresented groups and a slightly larger percentage with free or reduced lunch (over 40%). Similar to Ada, Hopper provides its students a relatively high degree of exposure to computing due to its STEM focus, enhanced even further by its close proximity to an engineering university.

The participating students stayed after school each day for two weeks, during which they played through the entirety of the ENGAGE game-based learning environment. We accepted anyone who wished to participate, but we placed an emphasis on attracting female students in order to have a relatively even gender split. A total of 18 students ended up participating in this after-school activity: 10 male students and 8 female students. There were 11 white students, 3 African-American students, 2 Asian students, and 2 Latino students. We see here an over-representation of white students from a school population in which about two-thirds of students are non-white, and this is consistent with study results of other out-of-school initiatives [17]. Furthermore, all but one of the after-school participants reported "yes" to the survey item, "Have you ever participated in any activities that involve computer science or computer programming?"

In other words, although we attracted a participant pool that contained a fair representation of female students, this after-school activity failed to provide the diversity of participant pool that could help us understand how the game-based learning environment impacts students who have no pre-existing interest in computer science. Unsurprisingly, these students all reported positive computer science attitudes, with an overall 3.89 (SD = .49). Indeed, these after-school participants reported positive computer science attitudes even in comparison to the in-school students with prior programming experience (reporting an average of 3.64, as seen above in Table 1). These differences in computer science attitudes get at the root of the problem we face with the out-of-school study implementation. Students who participate in these types of activities may already have an above-average interest in computer science. For an initiative such as ours that aims to reach a broad population of students, these out-of-school study implementations may not provide valid and generalizable assessments of how well our initiative affects students' learning and attitudes.

5. CONCLUSION

This paper highlights an area of need in K-12 computer science education research: in-school initiatives. Many current outreach initiatives involve some degree of self-selection. In the case of out-of-school initiatives, such as summer camps or after-school clubs, participants tend to already feel motivated to study computing. Further, these activities often place demands on the participants' families (e.g., transportation to and from the activity) that may disadvantage students from underrepresented backgrounds. Innovative curricula for computer science electives at the secondary school level address this issue of access, but they still likely enroll students who enter with higher computer science attitudes than the overall population. Furthermore, some evidence suggests that secondary school may be too late for significantly impacting computer science attitudes. To fully support the goal of broadening participation in computing, we must create in-school initiatives at the pre-secondary level.

Merely looking at demographic information (e.g., gender) does not provide enough information as to whether an initiative is reaching a truly broad population of students. We must reach out to students who would not otherwise consider computer science as a subject of study, in addition to nurturing the computing careers of underrepresented students who already have pre-existing interest in the field. Through research on initiatives with this goal, we can gain insight into how best to design learning activities that appeal to young students with little incoming computer science experience and low predisposition for computer science interest.

In future work, it is important to investigate how the various areas of development (teacher professional development, curriculum, and purpose-built software) interact with one other to produce the

most effective initiatives. A key issue is sustainability. While many projects have reported on the immediate results of an initiative, we need to follow up to see whether, and how, teachers are continuing in successive years. Finally, as the overarching goal is to recruit underrepresented students into computing, it will be critical to conduct longitudinal studies to measure the extent to which these in-school outreach initiatives empower students to study computer science at advanced levels.

6. ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation through Grants CNS-1453520 and CNS-113897. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

7. REFERENCES

[1] Aritajati, C. et al. 2015. A Socio-Cognitive Analysis of Summer Camp Outcomes and Experiences. *Proceedings of SIGCSE '15*, 581–586.

[2] Arpaci-dusseau, A. et al. 2013. Computer Science Principles : Analysis of a Proposed Advanced Placement Course. *Proceedings of SIGCSE '13*, 251–256.

[3] Bell, S. et al. 2014. Spreading the Word: Introducing Pre-Service Teachers to Programming. *Proceedings of SIGCSE '14*, 187–192.

[4] Bennett, V.E. et al. 2013. Computing Creativity: Divergence in Computational Thinking. *Proceeding of SIGCSE '13*, 359–364.

[5] Buffum, P.S. et al. 2015. A Practical Guide to Developing and Validating Computer Science Knowledge Assessments with Application to Middle School. *Proceedings of SIGCSE '15*, 622-627.

[6] Buffum, P.S. et al. 2014. CS Principles Goes to Middle School: Learning How to Teach "Big Data." *Proceedings of SIGCSE '14*, 151–156.

[7] Buffum, P.S. et al. 2015. Leveraging Collaboration to Improve Gender Equity in a Game-based Learning Environment for Middle School Computer Science. *RESPECT - IEEE Special Technical Community on Broadening Participation* (2015), 1-8.

[8] Carter, E. et al. 2012. Bringing the Breadth of Computer Science to Middle Schools. *Proceedings of SIGCSE '12*, 203-208.

[9] Franklin, D. et al. 2011. Animal Tlatoque: Attracting Middle School Students to Computing Through Culturally-Relevant Themes. *Proceedings of SIGCSE '11*, 453–458.

[10] Goode, J. and Margolis, J. 2011. Exploring Computer Science. *ACM Transactions on Computing Education*. 11, 2 (Jul. 2011), 1–16.

[11] Gray, J. et al. 2015. A Mid-Project Report on a Statewide Professional Development Model for CS Principles. *Proceedings of SIGCSE '15*, 380–385.

[12] Kay, J.S. et al. 2014. Sneaking in Through the Back Door: Introducing K-12 Teachers to Robot Programming. *Proceedings of SIGCSE '14*, 499–504.

[13] Koh, K.H. et al. 2013. Will it Stick ? Exploring the Sustainability of Computational Thinking Education Through Game Design. *Proceedings of SIGCSE '13*, 597-602.

[14] Lent, R.W. et al. 1994. Toward a Unifying Social Cognitive Theory of Career and Academic Interest, Choice, and Performance. *Journal of Vocational Behavior*. 45, (1994), 79–122.

[15] Levy, R.B.-B. and Ben-Ari, M. 2007. We Work So Hard and They Don't Use It: Acceptance of Software Tools by Teachers. *Proceedings of the 12th Conference on Innovation and Technology in Computer Science Education - ITiCSE '07*, 246-250.

[16] Liu, J. et al. 2014. Making Games a "Snap" with Stencyl – A Summer Computing Workshop for K-12 Teachers. *Proceedings of SIGCSE '14*, 169–174.

[17] McGill, M.M. et al. 2015. Does Outreach Impact Choices of Major for Underrepresented Undergraduate Students? *Proceedings of ICER '15*, 71–80.

[18] Ni, L. 2009. What Makes CS Teachers Change?: Factors Influencing CS Teachers' Adoption of Curriculum Innovations. *ACM SIGCSE Bulletin*. 41, 1 (Mar. 2009), 544–549.

[19] Oliveira, O.L. et al. 2014. Quantitative Correlation Between Ability to Compute and Student Performance in a Primary School. *Proceedings of SIGCSE '14*, 505–510.

[20] Pasternak, A. and Vahrenhold, J. 2012. Design and Evaluation of a Braided Teaching Course in Sixth Grade Computer Science Education. *Proceedings of SIGCSE '12*, 45–50.

[21] Rodger, S. et al. 2012. Integrating Computing into Middle School Disciplines through Projects. *Proceedings of SIGCSE '12*, 421–426.

[22] Schanzer, E. et al. 2015. Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap. *Proceedings of SIGCSE '15*, 616–621.

[23] Schofield, E. et al. 2014. MyCS: CS for Middle-Years Students and their Teachers. *Proceedings of SIGCSE '14*, 337–342.

[24] Seiter, L. 2015. Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of SIGCSE '15*, 540–545.

[25] Shashaani, L. 1994. Gender Differences in Computer Experiences and Its Influence on Computer Attitudes. *Journal of Educational Computing Research*. 11, (1994), 347–367.

[26] Tashakkori, R.M. et al. 2014. Research Experience for Teachers: Data Analysis & Mining, Visualization, and Image Processing. *Proceedings of SIGCSE '14*, 193–198.

[27] Webb, D.C. et al. 2012. Toward an Emergent Theory of Broadening Participation in Computer Science Education. *Proceedings of SIGCSE '12*, 173–178.

[28] Webb, H.C. 2011. Injecting Computational Thinking Into Career Explorations for Middle School Girls. *2011 IEEE Symposium on Visual Languages and Human Centric Computing VLHCC*. (2011), 237–238.

[29] Wiebe, E.N. et al. 2003. *Computer Science Attitude Survey*. Dept. of Computer Science, North Carolina State University Technical Report TR-2003-1.

[30] Yadav, A. et al. 2011. Introducing Computational Thinking in Education Courses. *Proceedings of SIGCSE '11*, 465-470.

[31] Yoo, D. et al. 2011. WeScheme: The Browser Is Your Programming Environment Danny. *Proceedings of ITiCSE '11*, 163-167.