Student engagement is key to broadening participation in CS

Beryl Hoffman
College of Our Lady of the Elms

Ralph Morelli
Trinity College Hartford

Jennifer Rosato
College of St. Scholastica

Student Engagement is Key to Broadening Participation in CS

Beryl Hoffman
Elms College
Chicopee, MA
hoffmanb@elms.edu

Ralph Morelli
Trinity College
Hartford, CT
ralph.morelli@trincoll.edu

Jennifer Rosato
College of St. Scholastica
Duluth, MN
jrosato@css.edu

ABSTRACT

The Mobile CS Principles (Mobile CSP) course is one of the NSF-supported, College Board-endorsed curricula for the new Computer Science Principles AP course. Since 2013, the Mobile CSP project has trained more than 700 teachers, and the course has been offered to more than 20,000 students throughout the United States. The organizing philosophy behind the Mobile CSP course is that student engagement in the classroom is the key to getting students, especially those traditionally underrepresented in CS, interested in pursuing further study and careers in CS. The main strategies used to engage Mobile CSP students are: (1) a focus on mobile computing throughout the course, taking advantage of current student interest in smartphones; (2) an emphasis on getting students building mobile apps from day one, by utilizing the highly accessible App Inventor programming language; and (3) an emphasis on building creative, 'socially useful' apps to get students thinking about ways that computing can help their communities. In this paper we present and summarize two years of data of various types (i.e., student surveys, teacher surveys, objective assessments, and anecdotal reports from students and teachers) to support the hypothesis that engagement of the sort practiced in the Mobile CSP course not only helps broaden participation in CS among hard-to-reach demographics, but also provides them with a solid grounding in computer science principles and practices.

CCS CONCEPTS

• **Social and professional topics → Computing education**; • **Software and its engineering →** *Visual languages*;

KEYWORDS

Mobile CSP, AP CSP, App Inventor, Broadening Participation, Student Engagement

ACM Reference Format:
Beryl Hoffman, Ralph Morelli, and Jennifer Rosato. 2019. Student Engagement is Key to Broadening Participation in CS. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287438

1 INTRODUCTION

Student engagement, defined as "the student's active participation in academic...activities, and commitment to educational goals and learning" [4], is essential to learning. Engagement can be measured along four dimensions: behavioral, affective (or emotional), academic, and cognitive [1]. In this study we are focused on cognitive engagement which has been defined as "thoughtfulness and willingness to exert the effort necessary to comprehend complex ideas and master difficult skills" [11]. Cognitive engagement can be readily observed by teachers and self-reported by students. Students often feel cognitively engaged in activities they perceive to be relevant to their lives and that have the potential to impact their communities.

While it is possible to have strong student engagement in a lecture-based classroom, in the current K-12 environment it is typically associated with active learning approaches which require the student to participate in the learning process. Active learning classrooms employ a core of pedagogical approaches, including collaborative, cooperative, inquiry-based, and problem-based learning strategies. NCWIT's Engagement Practices include a number of strategies for computer science educators to use both within and outside the classroom [8].

In computer science (CS) education, active learning can be traced back to the work of Seymour Papert. In the late 1960s, Papert and his colleagues [22] created Logo, the first programming language that was designed with student engagement and student learning in mind. It was Papert's revolutionary view that students should be learning to program computers rather than being controlled by them. This emphasis on creative student-initiated projects is a basic tenet of the Papert's contructionist approach to education, which views learning as an active process of building mental models and physical artifacts [17].

In this paper we describe research results derived from the past two years of study of the attitudes and achievements of students and teachers participating in the Mobile CS Principles (Mobile CSP) course, a CS Principles course deliberately designed to foster student creativity and engagement in the spirit of Papert.

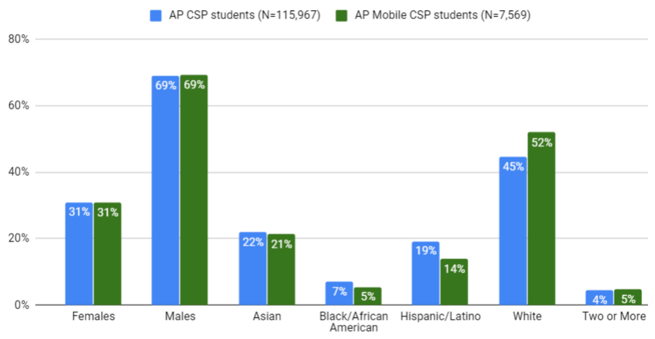
1.1 AP CSP

The Advanced Placement (AP) CSP curriculum is roughly equivalent to an introductory college CS0 course for non-majors. It offers a broad overview of CS, focusing on seven big ideas: abstraction, algorithms, programming, data, the Internet, creativity, and the social impact of computing [2]. It emphasizes cooperation, collaboration and project-based creativity. It was specifically designed with the goal of broadening participation in computer science both in terms of the number of students taking high school CS courses as well as the increased inclusion of underrepresented groups, such as women and minorities [5].

The AP CSP exam was officially launched in May 2017 with 50,000 examinees, breaking the record for the largest roll out of a new AP course; in the second year of 2018, more than 75,000 took the exam [10]. A unique part of the AP CSP exam is that students submit two performance tasks that are included in their score as well as taking a multiple-choice exam: the Explore task, a research project on a computing innovation, and the Create task, a programming project in any programming language. A key feature of the AP CSP framework is that it does not prescribe a programming language (in contrast to the AP CS A course, which is based on Java). Curriculum providers and teachers are free to use any programming language. For the purposes of the written exam, pseudocode and generic block code are used to express programming and algorithm questions and solutions. In the two years of the exam, a wide range of languages have been used by students to complete the performance task, including both blocks-based languages (App Inventor, Alice, Scratch) and text-based languages (JavaScript, Python).

The 2017 and 2018 AP CSP course broadened participation among female and underrepresented minority students compared to the AP CS A course. However, there is still much work to be done to increase these percentages which still leave out many female and minority students as can be seen in Figure 1. Mobile CSP students who took the AP exam had similar demographics to the total AP CSP students among all CSP curricula. Interestingly, Mobile CSP does have a higher percentage of minorities in its course registration which includes students who are not taking it as an AP course (20% Latinx, 11% African-American, for 20,000+ students).

Figure 1: Demographics of 2017 and 2018 AP CSP (n=115,967) and Mobile CSP Students (n=7,569)



1.2 Mobile CSP

The Mobile CSP course [6, 14, 20] is one of the NSF-supported, College Board-endorsed curricula for the new Computer Science Principles AP framework. Mobile CSP began as a college CS0 course in 2011 and then joined the College Board's Phase 2 pilot program. With support of a 2013 NSF grant, the course became the vehicle for training local teachers in summer professional development (PD) courses. With support of a second NSF grant in 2014, the project was expanded nationally with both local and online PD modalities. Since then, the Mobile CSP course has been used to train more than 700 teachers and it has been offered to more than 20,000 students throughout the United States.

The Mobile CSP course uses creative mobile app-building projects as a fundamental element of the learning experience, giving students the opportunity to create solutions to problems they care about. The curriculum consists of a set of 80-100 45 minute lessons with a mixture of coding and CS theory taught in a hands-on interactive fashion.[1] About half of the course's lessons focus on programming and algorithms (i.e., app building). Programming lessons use a combination of three approaches: video and text tutorials to introduce new concepts, small projects where students use pair programming to solve open-ended problems, and creative projects where students are encouraged to express their own ideas. The other half of the lessons covers non-programming CS Principles (CSP) topics, such as data, the Internet, and computing impact. Table 1 provides an overview of the types of lessons in the course. Multiple modes of learning are provided such as video tutorials, text tutorials, guided-inquiry group activities based on POGIL (Process Oriented Guided Inquiry Learning) [18], and interactive simulations. While both programming and non-programming lessons take an active-learning approach, our data (see Table 4) show that app-building lessons clearly provide the motivational "hook" that gets students engaged in learning challenging concepts.

Table 1: Types of Mobile CSP Lessons

Lesson Type	Number of 45 min lessons
App coding	26
CSP topics	24
CSP POGIL topics (group work)	10
Impacts of CS	11
AP Create Tasks (app coding)	24
AP Explore Tasks (impacts)	16

With its emphasis on mobile app building the Mobile CSP curriculum provides a robust model for engaging a diverse population of students to improve their skills and attitudes towards computer science. The course's organizing philosophy is that student engagement in the classroom is the key to getting students, especially those who are traditionally underrepresented in CS, interested in pursuing further study and possible careers in CS. The main strategies used to engage students are a focus on mobile computing and its social impact throughout the course, taking advantage of current interest in smartphones; an emphasis on getting students building mobile apps from Day 1; and an emphasis on building creative, "socially useful" apps that solve real-world problems. Learning to program by learning to program mobile apps is an example of contextualized learning, similar to media computation which has been shown to be effective in motivating underrepresented students to learn CS within a context they are interested in [9].

To focus on mobile computing, the Mobile CSP curriculum uses App Inventor as the programming language. Created originally by Hal Abelson of MIT during a sabbatical at Google, App Inventor was deliberately designed in the Papertian tradition [16, 25, 26].

[1]The curriculum is freely available at course.mobilecsp.org.

Like other blocks-based languages, such as Scratch and Alice, App Inventor's use of blocks makes programming accessible to complete novices. Block languages have been shown to improve learning for all students, and especially for those traditionally underrepresented in CS. In a comparative study of over 5,000 students answering block-based and text-based questions, researchers [24] found that all students performed better when questions were presented in the block-based form, with female and underrepresented minority students showing the largest improvements.

The significance of App Inventor's accessibility and computational power cannot be overstated. With App Inventor, students are able to create mobile apps from Day 1 of the course – i.e., apps that work the same way as apps that they are familiar with on their mobile devices. The language provides access to all of the device's hardware and communication features.[2] For example, one of the lessons in the course focuses on using the mobile device with its camera or voice recognition or text-to-speech generation. Unlike most other blocks languages, App Inventor's programming abstractions are rooted to actual mobile devices. The degree of abstraction in App Inventor components makes working on real world problems more feasible, providing functions such as sprite animation, text to speech generation, speech recognition, text translation, databases, maps, etc.

For the AP CSP's Create Performance Task, students design their own mobile apps and are encouraged to make them "socially useful". Some of the apps that students have designed are quite ambitious, and several of these have been used outside the classroom in student-initiated extra-curricular projects, demonstrating a high level of engagement. For example, one group of four Hartford high school girls worked with the local police department to create the Emergency Messenger App, a crowd-sourcing app to report crimes and suspicious activity in their neighborhood, and won first prize in the first annual Congressional App Challenge (previously named the Congressional STEM Competition) [19]. Many teachers report that they organize show-and-tell activities in their schools where students can display their apps to parents and school officials. For example, each spring a Mobile Apps Expo is held at Trinity College where students from area high schools demo their apps. Several of the apps that began as performance tasks have gone on to win regional or national contests [23].

An emphasis on building "socially-useful" apps is essential to getting students thinking about ways that computing technology can help their communities in classrooms, families, and neighborhoods. Students learn better when their classroom experience is connected to their lives [3]. The nature of the projects will depend on the interests and abilities of the students themselves. We observe that even simple apps, such as quiz apps that help students study a particular subject or apps that do something beneficial for one's school, can engage students in creative, constructionist learning.

Taken together these strategies not only help broaden participation in CS among hard-to-reach demographics, but also provide students with a solid grounding in computer science principles and practices.

[2]App Inventor was originally designed for Android devices. But a beta release of the language now supports iOS (Apple iPhones). The Mobile CSP course has been adapted to use both Android and iOS platforms.

2 RESULTS

2.1 Methods

The Mobile CSP project provided professional development (PD) for 275 high school teachers in the summers of 2016 and 2017 for those teaching the course in the following academic years (2016-17 and 2017-18). The PD of approximately 100 hours included an introduction to the course, its lessons, and the pedagogical strategies used to teach the course. It was offered in hybrid or online formats; hybrid included two weeks of in-person instruction with two weeks of online instruction. The summer PD was led by master teachers, largely high school teachers who had previously taught the course and received training on offering PD. During the academic year, the master teachers also led monthly web conferences to discuss progress in the course and provide support to first-time teachers. The larger Mobile CSP community of teachers interacted through a discussion forum and had access to monthly webinars from the project team. The teachers trained in the PDs participated in the research project to gather student data in their classes. They administered pre and post surveys and a midterm and final exam to approximately 6,000 students in 2016-2018.

The following data was collected from teachers and students in the research study:

- Teacher pre-PD, post-PD, and post-course surveys,
- Student pre-course and post-course surveys,
- Student midterm and final exam scores
- Student Create and Explore performance task scores (graded by their teachers using the AP rubrics

Data was collected on student engagement through a survey administered to teachers after the course was completed. Students were given a pre-and post-course survey on their attitudes and interest in CS. Data on student performance was similar to the AP exam itself with scores for final exams and the create and explore performance tasks. A common final exam was used with auto-grading while teachers used the AP rubrics for each performance task to report a score to researchers.

The demographics of students in the Mobile CSP research study in the past two years is detailed in Table 2. As can be seen, the percentage of females at 29% is very close to the 31% rate participating in AP CSP. The underrepresented minorities (African-American, Hispanic/Latino, American Indian/Alaska Native, Native Hawaiian/Pacific Islanders) make up 32% of the students in the research study. We are not reporting the results for minorities for which there were too small numbers (American Indian/Alaska Native, Native Hawaiian/Pacific Islanders).

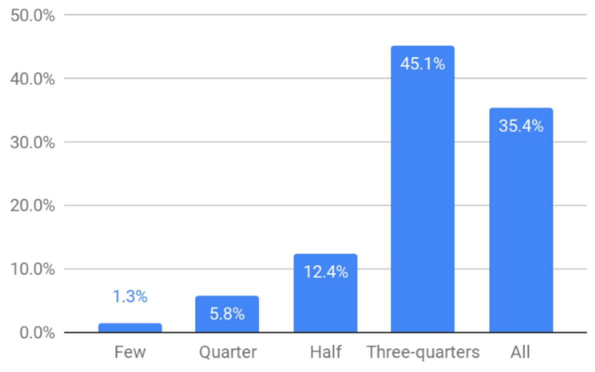
Table 2: Student Demographics in 2016-2018 Research (n=6492 in pre-surveys)

Demographics	Number	Percentage
Male	4609	71%
Female	1883	29%
Black	858	13%
Latino	1260	19%
Total URM	2118	32%

2.2 Engagement

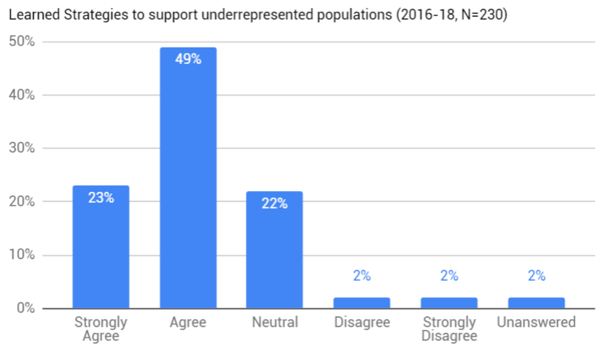
As described above, the project's data measures student cognitive engagement through a number a different instruments from both teachers and the students themselves. In post-course surveys, teachers report very high levels of engagement among students; nearly 80% of teachers reported that three-quarters or more of their students found the course "engaging" (Figure 2). Although the survey did not explicitly define "engaging" in terms of cognitive engagement, we believe it is the most relevant interpretation of the term given the context.

Figure 2: Teacher post-survey 2016-2018: What proportion of your class found the course engaging (n=226)



The Mobile CSP PD for teachers emphasizes pedagogy that engages students and broadens participation. In teacher post-surveys in 2016-2018, 72% of the teachers agreed or strongly agreed that the Mobile CSP program helped them to better understand how to implement strategies to support underrepresented populations in CS as seen in Figure 3.

Figure 3: Teacher post-survey 2016-2018: The Mobile CSP program helped me better understand how to implement strategies to support underrepresented populations in CS (n=230)



Teachers in the Mobile CSP PD are taught pedagogical strategies to improve student engagement and learning for all students, especially reaching those underrepresented and perhaps new to computer science. Strategies taught include growth mindset, pair programming, welcoming physical environment, project based learning, etc. Teachers are encouraged to use pair programming in class which has been shown to improve student learning, success and persistence [12, 13]. Physical environments that are not gender-biased are welcoming to all students [15]. Encouraging growth mindset in students leads to persistence [7].

In the 2016-2018 post-course surveys, teachers were asked how well they were able to implement the following pedagogies that they learned during the PD in teaching the course as seen in Table 3. Teachers were able to successfully use these strategies in their classes. With pair programming, some teachers report problems such as finding pairs that worked well together, and making sure both students did equal amounts of work.

Table 3: Teacher post-survey 2016-2018: How well were you able to implement the following strategies (n=230)

Worked	very well	well	some	no
Growth Mindset	20%	47%	23%	0%
Pair Programming	27%	27%	37%	4%
Welcoming Environment	47%	40%	8%	1%
Project based learning	47%	36%	4%	1%

Teachers in general report that App Inventor lessons are more popular and engaging with their students than the theory CS principles lessons, even with those involving group work, as seen in Table 4. This supports the view that students are engaged by lessons involving active learning that are contextually relevant and applicable to their lives and interests.

Table 4: Teacher post-survey 2017: How well did lessons work (n=147)

	Very Well	Well	Both
App Inventor Lessons (22)	42%	41%	84%
Other Lessons (30)	22%	49%	71%

2.3 Student Learning

Student engagement can also be demonstrated through student success in learning. In the first year of the AP CSP exam in 2017, 3,611 Mobile CSP students took the AP CSP exam with a pass rate of 78%, 4 points above the national average of 74% for passing the AP exam with a score of 3 or higher. In 2018, 4,031 Mobile CSP students took the AP CSP exam with a 76% pass rate, 7 points higher than the national average of 69%. Mobile CSP students met or exceeded the national average in every content area and performance task. In addition, Mobile CSP students out-performed the national average in every demographic group as seen in the Figure 4 for 2017 and in most groups as seen in Figure 5 for 2018. We were pleased to see that female and male students were very close in their success using the Mobile CSP curriculum. This data supports the view that using a curriculum like Mobile CSP which engages students with a variety of methods, including an accessible block language with highly abstract, useful blocks to create apps for problems that matter to them, may show better gains for underrepresented groups. However, there is still much room for improvement in both recruiting underrepresented groups (see Figure 1) and in ensuring their success (see Figure 4 and Figure 5).

Figure 4: Success of 2017 AP CSP and Mobile CSP Students who passed the AP CSP exam with 3 or higher

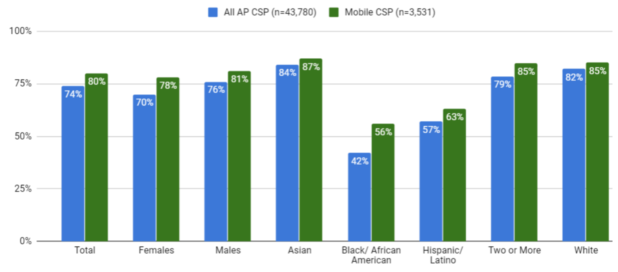


Figure 5: Success of 2018 AP CSP and Mobile CSP Students who passed the AP CSP exam with 3 or higher

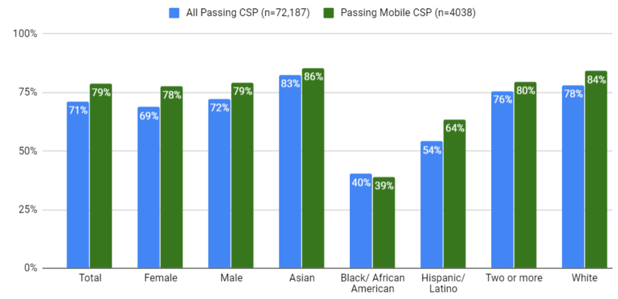


Table 5: Mobile CSP Student Average Grades in Class (2017 and 2018).

	All	Female	Black	Latino
Final Exam ave.	80%	83%	72%	74%
Create 2	84%	87%	74%	79%
Explore 2	70%	67%	56%	62%
Final Grade (2017)	82%	84%	75%	75%

Mobile CSP PD teachers participating in the research study are asked to give a standard Mobile CSP final exam and submit data about their students' success in their classes. Table 5 from 2016-2018 shows that female and underrepresented minority students did better in their Mobile CSP final exam, Create project, and their final grade in the class than on the AP exam. This makes sense due to the high stakes one day test for the AP exam which makes up the bulk of their AP score. Many students enjoyed and did well on the creative programming projects that are reflected in their course grade and performance task grades.

2.4 Student Attitudes

Although our student surveys have no questions that directly measure student engagement, there are several questions that indirectly speak to student engagement. In post-course surveys students were asked various attitudinal questions about their experiences in the course (Table 6). Large majorities of students (76%) reported that the course improved their understanding of computer science. Importantly, these results were consistent or higher across the targeted underrepresented groups, with 79% of female, 77% of Black/African-American, and 79% of Hispanic/Latino students reporting improved understanding. Similarly, more than two-thirds of students (66%) reported that the course improved their attitude toward computer science, which was consistent or higher across female and underrepresented minorities (68-72%). Students also reported an increased interest in CS (59%) and, importantly, that they discovered a new talent for programming (58%) with higher percentages for underrepresented minorities for whom this may have been a first experience in programming.

Table 6: Percent "quite true" or "completely true" on improvement in attitudes about CS after taking Mobile CSP 2017 (n=2218)

	All	Female	Black	Latino
The course improved my understanding of CS	76%	79%	77%	79%
The course improved my attitude towards CS	66%	68%	72%	72%
I learned that I have more programming talent than I was aware of	58%	61%	66%	65%
I have become more interested in CS	59%	56%	56%	66%

2.5 Student Future CS Plans

Strong student engagement is also suggested indirectly by student responses on questions focusing on their future plans. Table 8 presents a summary of gender vs. minority status with respect to future plans while Table 7 presents the full results for three different statements of increasing commitment to a future in CS: "I plan to take more CS in college", "I am considering a Computer Science major in college", and "I am considering a career in Computer Science and technology". In the 2017 and 2018 student post-surveys, nearly two-thirds (64%) of students expressed interest in either majoring in CS in college or pursuing a career in CS as seen in Table 8 which aggregates several questions. Even higher percentages (83%) answered Yes or Maybe to a statement "I plan to take more CS in college" as seen in Table 7. Female students did not feel as strongly committed to this plan as seen by more Maybe answers than Yes answers. Especially encouraging is the fact that there were no differences in the future CS plans of underrepresented minority students (African-American, Latino) compared with all the students. These results are strong evidence that the course is providing a positive view of computer science and helping to reinforce students' desire to continue their study of CS.

On the other hand, while 72% of the male students had future CS plans only 48% of female students reported such plans, suggesting that there is still work to be done in broadening CS participation. However, the fact that 48% of female students reported that they plan to major or work in CS is still a strong result considering that currently the computing workforce is only 26% female and less than 20% of CS/CIS majors are female [21].

Table 7: Student post-course survey 2017 and 2018: future CS plans (n=3024)

Plans for		All	Female	Black	Latino
More CS in college	No	17%	24%	23%	19%
	Maybe	37%	44%	40%	35%
	Yes	46%	32%	37%	45%
	YesOrMaybe	83%	76%	77%	80%
CS major	No	33%	49%	41%	36%
	Maybe	32%	29%	31%	29%
	Yes	35%	22%	28%	35%
	YesOrMaybe	67%	51%	59%	64%
CS career	No	31%	48%	42%	32%
	Maybe	33%	31%	27%	31%
	Yes	36%	22%	42%	37%
	YesOrMaybe	69%	52%	69%	68%

Table 8: Combining students considering or planning CS in college and/or career by URM and gender

Future CS Plans	In Total Students	In URM Students
Total	64%	62%
Female	48%	45%
Male	72%	72%

3 DISCUSSION

Our data suggest that the Mobile CSP course leads to a high level of engagement and learning in the classroom (and outside the classroom in some cases). Nearly 80% of teachers report that more than three-quarters of their students found the course engaging. Students from all demographic groups report increased interest in CS and improved attitudes towards CS. We believe the essential elements of student engagement in this course are: (1) a focus on mobile computing, which is important in students day-to-day lives; (2) the use of a programming environment (App Inventor) that makes it possible for students to create real world apps from the very beginning of the course; and (3) a requirement that students focus on apps that have some personal, community or social benefit.

Although one can't draw a direct causal connection between classroom engagement and success on the AP CSP exam, it certainly seems plausible to expect better performance from students that are more engaged in the way described here. While there is still an unacceptable lag in AP exam scores among females and underrepresented minorities, our data suggest perhaps that engagement can help diminish some of that gap. To help bridge these gaps a variety of extracurricular steps could be taken. More consideration could be given to the types of projects and apps that are developed to ensure they appeal broadly across genders and cultural groups. More attention can be paid to recruiting underrepresented students to bring them into closer alignment with school demographics. Efforts to promote positive role models and address implicit biases can help create a more welcoming classroom environment for all students.

As noted above, in our post-course surveys, students report strong interest in further CS study in college or as a career. The fact that the CSP course emphasizes creativity and project-based learning are strong incentives towards improving students attitudes towards CS. Even further, the type of hands-on engagement emphasized in the Mobile CSP course (and presumably other CSP courses) provides even stronger incentives. Our worry, however, is that students who experience computer science through the CSP or Mobile CSP lens will be disappointed and discouraged to find that not all introductory CS courses emphasize this type of engagement.

Hopefully some of the positive strategies and pedagogical lessons learned from the CSP experience can flow into the courses that follow CSP, such as in AP CS A or the CS1 and CS2 courses in college. That would ensure that students will find a similarly engaging environment, one that leverages their interests and active learning strategies and continues the trend of broadening participation in the CS pipeline.

4 ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under grants CNS-1240841 and CNS-1440947. We would also like to thank the Mobile CSP team for their dedication and contribution to this project.

REFERENCES

[1] James Appleton, Sandra Christenson, and Michael Furlong. 2008. Student engagement with school: Critical conceptual and methodological issues of the construct. 45 (05 2008), 369 – 386.

[2] College Board. 2017. AP Computer Science Principles: Course and Exam Description. Retrieved August 31, 2018 from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

[3] Lisa M Bouillion and Louis M Gomez. 2001. Connecting school and community with science learning: Real world problems and school–community partnerships as contextual scaffolds. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching* 38, 8 (2001), 878–898.

[4] Sandra L Christenson, Amy L Reschly, and Cathy Wylie. 2012. *Handbook of research on student engagement.* Springer Science & Business Media. 816–7 pages.

[5] Jan Cuny. 2015. Transforming K-12 Computing Education: AP&Reg; Computer Science Principles. *ACM Inroads* 6, 4 (Nov. 2015), 58–59. https://doi.org/10.1145/2832916

[6] Mobile CSP Curriculum. 2018. Course site: http://course.mobilecsp.org , Teacher site: http://teach.mobilecsp.org. Retrieved November 1, 2018 from https://mobile-csp.org/

[7] Carol S. Dweck. 2006. *Mindset: The New Psychology of Success.* Ballantine Books.

[8] NCWIT Engage CS Edu. 2018. Retrieved August 31, 2018 from https://www.engage-csedu.org/engagement/make-it-matter

[9] Mark Guzdial. 2015. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (2015), 1–165.

[10] Sasha Jones. 2018. Participation in AP Computer Science Principles Grows Again. Retrieved August 31, 2018 from http://blogs.edweek.org/edweek/curriculum/2018/06/by$_g$uest$_b$logger$_s$asha$_j$ones.html

[11] Duhita Mahatmya, Brenda J Lohman, Jennifer L Matjasko, and Amy Feldman Farb. 2012. Engagement across developmental periods. In *Handbook of research on student engagement*. Springer, 45–63.

[12] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin* 34, 1 (2002), 38–42.

[13] Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49, 8 (2006), 90–95.

[14] Ralph Morelli, Chinma Uche, Pauline Lake, and Lawrence Baldwin. 2015. Analyzing year one of a CS Principles PD project. In *Proceedings of the 46th ACM technical symposium on Computer science education*. ACM, 368–373.

[15] NCWIT. [n. d.]. How Does the Physical Environment Affect WomenâĂŹs Entry and Persistence in Computing? Retrieved August 31, 2018 from https://www.ncwit.org/resources/how-does-physical-environment-affect-women%E2%80%99s-entry-and-persistence-computing

[16] MIT News Office. 2010. The MIT roots of GoogleâĂŹs new software. Retrieved August 31, 2018 from http://news.mit.edu/2010/android-abelson-0819

[17] Seymour Papert and Idit Harel. 1991. Situating constructionism. *Constructionism* 36, 2 (1991), 1–11.

[18] CS Pogil. 2019. Retrieved August 31, 2018 from http://cspogil.org

[19] Press Release. 2014. Retrieved August 31, 2018 from https://esty.house.gov/media-center/press-releases/rep-elizabeth-esty-announces-congressional-design-your-own-app-contest

[20] Jennifer Rosato, Chery Lucarelli, Cassandra Beckworth, and Ralph Morelli. 2017. A Comparison of Online and Hybrid Professional Development for CS Principles Teachers. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 140–145.

[21] NCWIT By the Numbers. 2017. Retrieved August 15, 2018 from https://www.ncwit.org/sites/default/files/resources/btn$_0$4042018$_w$eb.pdf

[22] Paulo Blikstein (Stanford University). [n. d.]. Seymour Papert's Legacy: Thinking About Learning, and Learning About Thinking. Retrieved August 31, 2018 from https://tltl.stanford.edu/content/seymour-papert-s-legacy-thinking-about-learning-and-learning-about-thinking

[23] WCVB. 2016. 5 for Good: Local high School students win Congressional App Challenge. Retrieved August 31, 2018 from https://www.wcvb.com/article/5-for-good-local-high-school-students-win-congressional-app-challenge/8096710

[24] David Weintrop, Heather Killen, and Baker Franke. 2018. Blocks or Text? How programming language modality makes a difference in assessing underrepresented populations. In *Proceedings of the International Conference on the Learning Sciences 2018, At London, UK.*

[25] David Wolber. 2011. App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 601–606.

[26] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. 2011. *App Inventor*. " O'Reilly Media, Inc.".