Analyzing Year One of a CS Principles PD Project

Ralph Morelli
Computer Science Department
Trinity College
Hartford, CT 06106
ralph.morelli@trincoll.edu

Chinma Uche
Connecticut Computer Science Teachers Assoc.
Greater Hartford Academy of Math and Science
Hartford, CT 06106
cuche@crec.org

Pauline Lake
Computer Science Department
Trinity College
Hartford, CT 06106
pauline.lake@trincoll.edu

Lawrence Baldwin
Baldwin Institutional Research Consulting
2 Everett Street
Sherborn, MA 01770
lawrence.baldwin@verizon.net

ABSTRACT

The Mobile Computer Science Principles (Mobile CSP) project is an NSF-funded CS 10K project. Its goal is to train a new cohort of high school computer science teachers to teach an Advanced Placement (AP) level course based on the emerging Computer Science Principles (CSP) framework. Mobile CSP uses App Inventor, a graphical programming language for Android devices, to engage students in app building as a means to get them interested in computer science. This paper reports on the first year of this effort. In addition to describing the project's Professional Development (PD) course, a 6-week, full-time summer course for teachers, and the Mobile CSP curriculum, the paper provides a preliminary analysis of demographic and performance data obtained from high school students who took the course during the 2013-2014 academic year.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

1. INTRODUCTION

The Mobile Computer Science Principles (Mobile CSP) project is one of several projects funded by the National Science Foundation (NSF) in an effort to address the alarming lack of computer science course offerings at the high school level in the United States. Despite the enormous impact that computing has had on our lives and its growing importance in virtually every field of inquiry, scientific and otherwise, and despite the growing gap between available jobs in computing and high school and college graduates qualified to fill those jobs, U.S. high schools fail to offer rigorous courses in computer science. Connecticut is no exception.

As Table 1 shows, according to the College Board, only 460 CT students took the AP exam in Computer Science in 2013 as opposed to 4229 for AP Calculus AB and 3695 for Biology[5]. Even though Connecticut's numbers in CS have improved in the past two years, they still lag far behind the other STEM fields. This is a direct result of the lack of access to rigorous courses and well-trained CS teachers in Connecticut and elsewhere.

Table 1: Connecticut AP Data since 2011.

Year	2011	2012	2013
AP CS A	298	436	460
AP Calc AB	4046	4208	4229
AP Biology	3470	3331	3695
AP Chemistry	2165	2556	2573

As described in its 2011 solicitation, the goal of the NSF's CS 10K initiative is to increase the effectiveness of computing education in high school

> through the introduction of an entirely new curriculum (based on a proposed, new AP course) concomitant with the preparation of 10,000 high school teachers prepared to teach the new curriculum in 10,000 schools.[7]

The Mobile CSP project was funded in 2012 and addresses the CS 10K initiative through its three primary goals:

- To develop a core group of 30 CS teachers in Connecticut by training them in the Mobile CSP curriculum and supporting their efforts in the classroom.
- To create a complete set of online, openly licensed, curricular and instructional materials that can be disseminated broadly throughout Connecticut and elsewhere.
- To assess the efficacy of the curriculum and its impact on the attitudes and learning outcomes of participating teachers and their students.

SIGCSE'15, March 4–7, 2015, Kansas City, MO, USA.

http://dx.doi.org/10.1145/2676723.2677265.

The project has made good progress in addressing the first two of these goals. In 2013-14, the period covered in this report, 10 Connecticut teachers were provided formal CS training.[1] The training consists of a full-time, six-week summer Professional Development (PD) course.

With respect to the second goal, a complete set of Mobile CSP course materials are currently available online on two parallel *Google Course Builder (GCB)* sites: A teacher-facing site contains lesson plans, grading rubrics, solutions, and other supporting materials and a student-facing site[6] contains the course content.

This report provides an overview of the project, including its PD effort, and focuses primarily on the the efficacy of the Mobile CSP curriculum as measured by demographic and performance data gathered from students who took the course from the 2013 cohort of Mobile CSP teachers.

2. THE CS PRINCIPLES FRAMEWORK

CS Principles is a curricular framework being developed by the College Board for a new language-neutral, breadth-first AP course in computer science that is scheduled for release during 2016-17[1]. The framework is based on seven *big ideas*: 1) Creativity, 2) Abstraction, 3) Data, 4) Algorithms, 5) Programming, 6) The Internet, and 7) Impact.

These are expanded into 23 *enduring understandings*, which are further expanded into 42 specific *learning objectives* (i.e., what students need to be able to do), each of which is further expanded into more than 100 specific *essential knowledge* statements (i.e., what students need to know).

Table 2: Breakdown of a CS Principle

Big Idea	Abstraction
Enduring Understanding	A variety of abstractions built upon binary sequences can be used to represent all digital data.
Learning Objectives	Describe the variety of abstractions used to represent data.
Essential Knowledge	The interpretation of a binary sequence depends on how it is used.

As Table 2 shows, the CS Principles is a *framework*. It *describes* goals for student learning but does not *prescribe* a specific curriculum. This is especially true in the area of programming. Unlike, the CS A course, which requires Java, no programming language is specified for the CS Principles.

The proposed CS Principles course has a unique way of assessing student performance. It supplements the customary AP written exam, with in-course *performance tasks* [1], which count for 50% of the student's grade. Currently, there are two required performance tasks: The *Explore* task requires the student to identify and research a computing innovation that has had a signficant impact on some population and produce written responses to a set of specific prompts as well as a digital artifact (e.g., a video audio, etc.) "to express the effects of your chosen innovation." The *Create* task requires the student to work collaboratively with another student on designing, implementing, and presenting a computer program of the students' choice. These tasks are allotted 8 and 12 hours of class time respectively.

[1]Sixteen additional Connecticut teachers were provided formal training in summer 2014, but data from the 2014 cohort is not included in this report. A final cohort of Connecticut teachers will be trained in summer 2015.

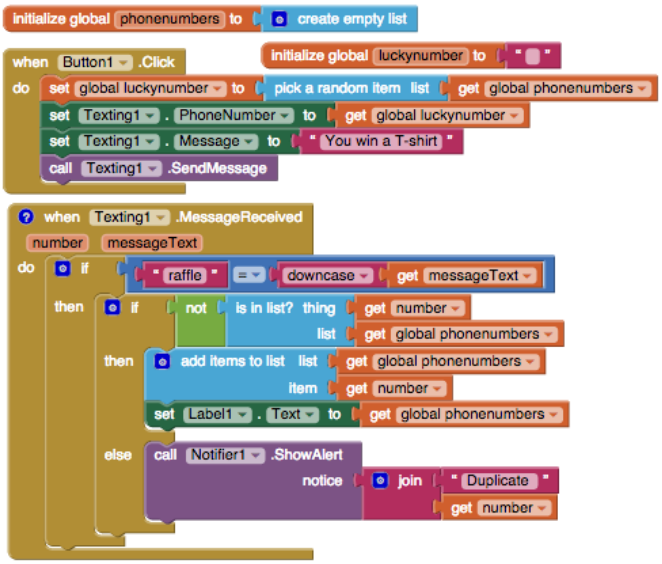

Figure 1: The App Inventor *Raffle* app.

3. APP INVENTOR

The Mobile CSP course uses App Inventor, a blocks-based visual language for programming Android mobile devices[3]. App Inventor is the invention of MIT Professor Hal Abelson and is hosted at the MIT Center for Mobile Learning.

The current version of App Inventor, which was released in September 2013, has more than 700,000 registered users who have created almost 2 million projects. Its usage data indicates that it is growing at a rate of roughly 2,500 new users and 6,500 new projects a day[2].[2] There are almost twice as many active daily users during the schoool year (13,400) as during the summer months (7,600), which suggests that App Inventor is becoming widely used in schools.

What makes App Inventor such an attractive language for introductory CS is its focus on mobile app development and its highly abstracted design. Programs (i.e., apps) are developed in a web browser and downloaded onto Android mobile devices either through a Wifi or USB connection. The download process can occur through *packaging*, in which the entire app is compiled into an Android APK file, or *interactively*, which permits iterative development and testing of the app.

App Inventor is *event-driven* and apps respond to a sequence of events, including User Interface (UI) events, such as button clicks, sensor events, such as orientation and GPS location changes, and communication events, such as phone calls and text messages. The language's blocks are mapped directly to the device's various hardware and software components and to its event hierarchy, making it quite simple for even novice programmers to design and implement sophisticated mobile apps.

App Inventor's powerful abstraction of mobile computing changes the basic formula for teaching introductory programming. Instead of saying "bear with us for a few weeks or months and you will soon be building some cool programs", an App Inventor course starts with building a mobile app on

[2]This growth likely includes users moving from the previous version, which had more than 1.5 million registered users.

the first day, in a matter of minutes. Students can quickly see how to express and implement some of their app ideas. The creative payoff is almost immediate and it is easy to show one's friends.

For example, one app that we have used as a day-one demo is the *Raffle* app. Students text the word "raffle" to the instructor's phone or tablet from any phone that supports SMS and the app registers them for a raffle. Once everyone in class is registered, the instructor presses a button and the app selects a random phone number and sends a "Congratulations, you've won!" message. This app consists of two pieces of code, one to handle the button-click event and one to handle text-message-received events (Figure 1) and can be developed in class in 10-20 minutes.

4. THE MOBILE CSP COURSE

The Mobile CSP course began as a CS0 "Mobile Computing" course at Trinity College in Spring 2011. In the following academic year, it was revised to conform to the CSP curricular framework and it was offered as one of the College Board's Phase 1 Pilot 2 courses, during which it was taught simultaneously at Trinity College and at a local public high school, the Greater Hartford Academy of Math and Science. The current version of the course attempts to conform to the CSP curricular framework layed out by the College Board in its most recent documents[1].

The course is currently organized into seven units and consists of more than 60 lessons, 28 of which are App Inventor lessons[6]. The remaining CSP lessons cover such topics as binary numbers, sorting and searching algorithms, cryptography, and other CS topics. The course includes an extensive collection of interactive *formative assessment exercises* – a collection of questions and exercises that accompany each lesson, including both App Inventor and CSP lessons. The exercises, which come with extensive hints and feedback, are not graded and can be repeated as many times as necessary to get the right answer.

The course takes the approach that creating mobile apps is the motivating factor through which students are introduced to computer science. Programming (app building) is seen as a creative process that requires students to learn some computer science in order to do well.

4.1 Mobile CSP PD

The project's PD consists of an intensive full-time 6-week course offered in the summer. Participants work through the entire Mobile CSP curriculum, from the same perspective that their students will later take. The curriculum includes app-building lessons and there is one lesson in each unit that focuses on readings and discussions of impact topics from *Blown to Bits*[4]. The remaining lessons are on CS topics ranging from the binary number system, to hardware and software abstractions, to algorithm analysis.

PD participants also complete scaled down versions of the CSP performance tasks. In summer 2013, there were three performance tasks, a programming task, a data analysis task, and a research project on the Internet.[3] Because most particants are new to programming – and especially App Inventor programming – we provide additional *creative projects*, which are scaled down versions of the programming performance tasks. Teachers work in pairs on all performance tasks and programming projects and are required to produce a write up and an oral presentation of their work.

Like their students are required to do for the course itself, the teachers maintain Google sites portfolios of their work. These include their personal reflections on the *Blown to Bits* readings, their reflections, as teachers, on the various lessons, and their write ups and documentation of the various performance tasks.

4.2 In-Class Support for Teachers

A very important component of the project's PD is its use of an *itinerant teaching consultant* who conducts regular in-class visits to provide moral support for new teachers and help them with lessons, equipment, and any other issues that come up while teaching the course the first year.[4] In addition to in-class support, teachers also have access to a small online community that reponds quickly to questions on a forum and holds bi-weekly Google hangouts and bi-monthly CSTA meetings to talk about the course and any issues that come up.

4.3 The 2013 Mobile CSP Teacher Cohort

In summer 2013, 10 Connecticut teachers participated in the Mobile CSP PD. As a condition for participation both the teacher and their principal signed an agreement that they would teach the Mobile CSP course in their school during the 2013-14 academic year. Teachers received a $6,000 stipend for active participation in the PD, plus another $1,000 stipend if they submitted all of the required data from their students during the year. Of those 10 teachers, 4 were certified in Technology Education, 2 in Social Studies, 2 in Math or Business, 1 in Science and 1 in Art. Three of the teachers had some prior experience teaching programming, one of whom had taught the AP CS A course. Seven of the 10 reported having no prior CS experience.

5. ANALYZING YEAR ONE

5.1 The 2013-14 High School Courses

During the school year, 9 of the 10 teachers taught some variation of the Mobile CSP course. They were joined by several other teachers who did not participate in the PD but who had prior CS experience and asked to participate in the project. A total of 17 sections of the course were taught, with some teachers teaching multiple sections. Seven teachers taught the course as a full-year course, the preferred model. Four taught it as a one-semester course in the fall and two taught it as a one-semester course in both the fall and spring semesters.

Seven of the 10 PD teachers made a good-faith effort to teach the full Mobile CSP course. Three of those succeeded, more or less, in doing so. The other 4 ended up dropping certain lessons or activities for various reasons. Two of the

[3]The College Board has revised the performance tasks. There are currently just two, the data analysis task has been dropped and the Internet task has been broaded to focus on a computing innovation in any area.

[4]The current consultant, author 3, is a 2013 graduate of Trinity College, who double majored in CS and Education and began teaching App Inventor to high school students while still an undergraduate.

PD teachers used only the App Inventor lessons in their courses and one teacher did not teach a course.[5]

5.2 Teacher Confidence and Success

At the end of the school year, 10 teachers completed a survey, 9 of the PD teachers and 1 additional teacher, that consisted of questions about their attitudes towards the course and there preparation for it, as well as their opinions about how each of the lessons went in the classroom.

Of the nine PD teachers surveyed, seven reported that they were very confident (3) or generally confident (4) that the course they were about to teach would go well. Two reported that they were not very confident. At the completion of the school year the 10 teachers who participated in the survey were asked how they liked the course after they had taught it. Five reported that they liked it "very much", four that they liked it "pretty well" and one reported that "it is ok". None reported "not well."

5.3 Analysis of Mobile CSP Lessons

The 2013-2014 version of the Mobile CSP course contained 65 lessons, 27 App Inventor lessons and 38 CSP lessons. The teachers' collective assessment of these lessons are summarized in Table 3.

Table 3: Success with Mobile CSP Lessons

	High	Medium	Low	Unused
Tutorials	72.7%	9.7%	3.2%	14.2%
Mini Projects	88.6%	1.9%	50%	9.6%
Creative Projects	79.3%	0%	0%	20.6%
CSP Lessons	58%	18.9%	0.5%	22%

The 27 App Inventor lessons included 16 *tutorials*, where the student follows a step-by-step lesson that teaches how to build an app, and 11 *mini projects*, where students are given problems and work in pairs to solve them on their own. The problems frequently involve enhancements to the app that was built in the tutorial.

In addition, there were three *creative projects*. These were modeled on the College Board's programming performance task. Students, working in pairs, proposed their own project within certain general constraints – e.g., the app must have a socially useful purpose and must make use of GPS. They then designed and implemented their app, wrote it up on their portfolios, and gave an oral presentation on it to the class.

As Table 3 shows, the App Inventor portion of the course was highly successful in the teachers' opinion. Indeed, if we discount the "Unused" column and count only those opinions among teachers who did the lesson, then the percentage of those saying the lessons were highly successful would be 84.8% for tutorials, 97.8% for mini projects and 100% for creative projects. This is a strong result from the teachers.

The row classified as CSP lessons are all the lessons that did not directly involve App Inventor. As Table 3 shows, the CSP portion of the course did not work as well for teachers as the App Inventor portion, with only 58% percent saying the lessons were highly successful and 18.9% reporting that there were problems in delivering the lessons. Note also that 22% reported "Unused." Indeed, as noted earlier, a couple of teachers seemed to have omitted the CSP portion of the curriculum entirely.

This difference between the App Inventor and non-App Inventor parts of the course was not unexpected and is consistent with the premise that mobile app building would be what attracts teachers and students to the course. As we will see shortly, this same difference arises among student preferences.

5.4 Student Data

The project collected various demographic and performance data from students, including: (1) pre- and post-course surveys of demographic data and attitudes toward CS; (2) mid-term and final examinations that covered both App Inventor and CSP topics through a combination of multiple choice, fill-in and open ended questions; and (3) performance tasks that were graded by teachers using a rubric provided by the project. Of the 354 students who took the Mobile CSP course at 10 schools we were able to collect pre- and post-survey data from 281 students and final exam data from 264 students.

Table 4: Student Demographics, including gender (M/F) and underrepresented (U)

Teacher	UF	F	UM	M	Tot
A	2	9	8	71	90
B	0	0	5	2	7
C	0	14	0	25	39
D	6	5	5	7	23
E	0	1	1	8	10
F	2	4	5	16	27
G	7	0	11	3	21
H	8	2	15	16	41
I	3	1	4	0	8
J	0	0	3	9	12
Totals	28	36	57	160	281
Totals	64		217		

5.5 Demographic Profile of the Students

The pre-course survey asked students for various demographic data, data about their prior experience, and their attitudes toward computing. As shown in Table 4, the male:female breakdown was 77% to 23%. Although the Mobile CSP course is designed for 11th and 12th AP-level, the students came from all grade levels: 9th (24%), 10th (29%), 11th (21%), and 12th (25%).

The racial/ethic breakdown was as follows: Caucasian (58%), African American (15%), Hispanic (14%), Native American, including Hawaiian and Alaskan (4%), Asian (10%), all others (10%).[6] For the purposes of this study an *underrepresented minorty* includes African American, Hispanic, Native American, including native Hawaiian and Alaskan, and girls. There was considerable variation in racial/ethnic backgrond and gender by school, as summarized in Table 4.

The vast majority of the students reported that they were experienced users of various computer applications, including email, Internet searches, social networking, games. But

[5]To try to minimize this type of attrition, we have tightened up our application process and our "contract" with the schools.

[6]Students were allowed to check more than one category so the total exceeds 100%.

Table 5: Change in Confidence

	Less	Same	More
Design something novel and innovative			
All Students	8%	46%	46%
Underrepresented	7%	47%	46%
Solve an unstructured problem			
All students	12%	38%	51%
Underrepresented	15%	43%	42%
Apply an abstract idea to a real problem			
All students	7%	38%	55%
Underrepresented	6%	39%	56%
Identify/fix a problem in a complex process			
All Students	12%	41%	47%
Underrepresented	15%	39%	47%
Work with others in small groups to solve problems			
All students	10%	29%	60%
Underrepresented	14%	28%	58%

relatively few reported prior programming experience; only around 30% of boys and less than 20% for girls. When asked to rate their programming ability at the start of the course, approximately 54% of all students rated themselves as "beginners", and only 11% of girls and 14% of boys described themselves as having "considerable experience" in programming.

5.6 Recruitment and Class Enrollment

Recruitment of students to the Mobile CSP course was a major challenge, especially within the Hartford school district, the primary target demographic. The project directors had little control over whether and how the course was advertised and promoted within the schools and it appears that in some cases it was not widely promoted among students or teachers. In most cases teachers were recruited and selected by the principal. Recruitment and selection of teachers occurred between January and April and in some cases the course was not put into the school's schedule until after students' schedules had been determined.

When asked in the pre-course survey why they took the course, 14% reported that it was recommended by a teacher and 52% said that the course sounded interesting or fun. But 41% also reported that they were assigned to the course.

5.7 Student Problem Solving Ability

Students were asked a number of questions in the pre- and post-course surveys about their confidence along various problem-solving dimensions. As Table 5 summarizes, their confidence as problem solvers seems to have improved overall and there was no significant difference observed for underrepresented minority students.

This is a positive result due, no doubt, to the emphasis and the amount of time spent in the course on designing and development of mobile apps. Moreover, their improved confidence seems to have been justified by the generally good work they did on their mini projects and on the creative performance tasks. For teachers who reported these grades to us, the overall average was 88%.

5.8 Student Attitudes Toward CS

The post-course survey revealed that 65% of students enjoyed the course as opposed to 12% who did not or 23% who were neutral. And 56% said they would probably take another CS course as opposed to 19% who said they would not and 26% who were unsure. Students were evenly divided (37%) on whether or not they might major in CS, with 27% neutral on that question.

Table 6: Attitudes Towards Computer Science

	Disagree	Maybe	Agree
Course improved my understanding of CS			
All students	2%	19%	80%
Underrepresented	1%	17%	22%
Course improved my attitude toward CS			
All students	8%	23%	69%
Underrepresented	2%	21%	77%
Learned I have some programming talent			
All students	5%	25%	69%
Underrepresented	5%	27%	69%
CS is a socially beneficial discipline			
All students	6%	21%	73%
Underrepresented	9%	29%	61%

Table 6 summarizes student responses to some of the questions that probed student attitudes toward various aspects of CS. Their attitudes were generally positive. On the other hand, the survey also revealed that we have more work to do on some of the standard misconceptions about computer science. More than half of the students (51%) agreed with the statement that *computer science is about using application software* and about the same number agreed (31%) and disagreed (29%) with the statement that computer science and programming are the same thing.

5.9 Student Final Exam Performance

The final exam for the course consisted of 40 multiple-choice, short-answer and short-essay questions based largely on the course's formative assessment exercises. Teachers were not given the exam before hand but were told that that students could prepare for the final by reviewing those exercises. We are not sure to what degree the students "studied" for the final. All but two of the teachers factored the final exam scores into the grades they gave their students.

As shown in Table 7 the average score for the 266 students who took the final was 62%, which was also the average for all underrepresented students. Student performance on the final varied considerably from teacher to teacher. There are a number of factors that could explain these differences but the two that stand out the most are (1) the level of the teacher's background and experience in CS and (2) whether the teacher covered all of the topics.

For example, teachers D and E had prior experience teaching CS and both covered the entire Mobile CSP curriculum. Their averages were above 80%. On the other hand teacher I was not experienced but did cover the whole curriculum and gave students time in class to study for the final. Teacher I also reported having received excellent in-class support from the project's itinerant teaching consultant.

By contrast, teachers C and G had prior teaching experience in CS. Teacher G, however, covered almost none of the CSP lessons and teacher C modified the recommended curriculum. Their average scores were in the 60s. On the other hand, teacher A, one of the more inexperienced teachers in the cohort, made a good faith effort to cover the entire curriculum but was unable to do so and was also unable to receive in-class support from the iterant teaching consultant. Student performance was very low (51%) in this case. We will need to take a look at additional data to try understand the reasons.

Table 7: Student Performance on Final Exam

Teacher	Score (%)	N-all
A	51	100
B	66	5
C	37	43
D	81	23
E	84	10
F	61	22
G	60	41
H	48	10
I	82	12
Totals	62	266

What do these results say about the PD and the final exam? Clearly, the exam is non-trivial: in classes where the material was not adequately covered, the students did not perform well. At the same time, in most courses where the material was covered, whether by experienced or inexperienced teachers, the students seemed to perform well. In general, student performance on the final seems to suggest that the PD and the in-class support provided by the project do have the potential to deliver AP-level CSP instruction.

6. DISCUSSION

The CS 10K initiative involves three very tough challenges: (1) Recruiting and training teachers with no formal education in CS to teach an AP-level CS course; (2) engaging students with no prior computing background in the study of AP-level computer science; and (3) getting school administrators to add rigorous computer science courses to their curricula in the absence of standards.

Training Teachers. Despite the relatively short training provided to teachers, the Mobile CSP approach seems to have worked at least to some extent. Despite their varied backgrounds, those who participated in the PD came away feeling confident in their ability to teach the course and nearly all who taught it described it as a positive experience.

Looking ahead, most of the 2013 cohort (8 out of 10) are teaching the course for a second year. One area for improvement is to work at making sure that teachers are teaching the full Mobile CSP curriculum.

Student Readiness. While students performed very well over all on the mobile app building projects, their performance on the final exam was not uniformly good. We think this was due to two factors. Some of the students were not ready for an AP-level course – 53% were 9th and 10th graders – and in several cases they were not exposed fully to the entire Mobile CSP curriculum. It is not clear to what degree students had the recommended prerequisites (Algebra I and Geometry) for the course.

This observation raises a broader question about the CS 10K initiative – namely, the goals of trying to offer an AP-level course as an introduction to computer science and also trying to attract an underrepresented demographic may be in tension with each other. This is not to suggest that one or the other of these goals should be dropped. On the contrary, both are worthy and necessary goals. It may suggest that there needs to be a pre-requisite computing course that students can take before taking the AP CSP course.

High School Curricula. We were unable to get 100% participation by the teachers who participated in the PD. One teacher ended up not teaching any course. Some focused mostly on the App Inventor lessons. Recruiting students, especially girls, was a big challenge. Some teachers reported not getting great help from guidance counselors. These challenges were exacerbated by the fact that the course was offered as an "elective," which lessons its attractiveness in the eyes of students and parents.

To address the variation in the "product", we have tightened up the "contract" between the project and the school. This seems to have improved the 2014 teacher cohort – at least in terms that they all agree that they are going to teach the full Mobile CSP curriculum. We also see the need to work with school boards and legislators to get CS accepted as a required course.

7. CONCLUSION

For the most part, the first year of the Mobile CSP PD effort has gone reasonably well.

- The intensive six-week course appears to provide teachers, even those with no prior CS background, with the confidence they need to teach the course.
- The lessons, lesson plans, and other pedagogical moterials seem to be well received by both the teachers and the students.
- The course itself appears to improve student attitudes toward CS and students appear to think that the course has helped improve their problem-solving skills.

These are positive results. Thus, despite the various problems mentioned above, these results do seem to support the project's underlying premise: mobile computing in App Inventor is a suitable and highly attractive means of getting students, including underpresented students, engaged in introductory CS education.

ACKNOWLEDGMENT

This work is supported by NSF Grant CNS-1240841.

8. REFERENCES

[1] Computer Science Principles, `http://csprinciples.org`, accessed Sep 4, 2014.

[2] App Inventor Usage Statistics, `http://ai2.appinventor.mit.edu/stats`, accessed Sep 1, 2014.

[3] The MIT Center for Mobile Learning, `http://appinventor.mit.edu`, accessed Sep 1, 2014.

[4] H. Abelson, K. Ledeen, and H. Lewis. *Blown to Bits: Your Life, Liberty, and Happiness After the Digital Explosion*. Addison-Wesley Professional, 2008.

[5] The College Board. AP Research Data, 2014. `http://research.collegeboard.org/programs/ap/data`, accessed Sep 5, 2014.

[6] The Mobile CSP Project. `http://mobile-csp.org`, accessed Sep 4, 2014.

[7] The National Science Foundation. Computing education for the 21st century, `http://www.nsf.gov/pubs/2010/nsf10619/nsf10619.txt`, accessed Sep 1, 2014.