Building Equitable Computer Science Classrooms: Elements of a Teaching Approach

Niral Shah
University of California, Berkeley
niral@berkeley.edu

Colleen M. Lewis
Harvey Mudd College
lewis@cs.hmc.edu

Roxane Caires
University of California, Berkeley
roxane.caires@berkeley.edu

Nasar Khan
University of California, Berkeley
nasarmkhan@gmail.com

Amirah Qureshi
University of California, Berkeley
amirahqureshi@gmail.com

Danielle Ehsanipour
University of California, Berkeley
d.ehsanipour@berkeley.edu

Noopur Gupta
University of California, Berkeley
noopurgupta@berkeley.edu

ABSTRACT
This paper offers a framework for equitable instruction that emerged while designing a computer science course for students entering the sixth grade. Leveraging research from a range of fields, including sociology, mathematics education, and the learning sciences, we argue that in addition to material resources, such as rich course content and quality instruction, equity also depends on students' access to non-material resources, such as productive domain identities and peer relationships. We illustrate each dimension of the framework by describing in detail a core set of pedagogical practices implemented during a summer course.

Categories and Subject Descriptors
K.3.2 **[Computers and Education]:** Computer and Information Science Education—*computer science education*

General Terms
Human Factors

Keywords
Scratch, equity, identity

1. INTRODUCTION
A primary goal of the National Science Foundation's *Computing Education for the 21st Century* (CE21) program is to identify pedagogical practices that may broaden participation among students from groups who are underrepresented in computer science, such as women and racial minorities. In fact, the CE21 initiative reflects an increasingly field-wide concern for equity, which is the idea that opportunities to learn computational thinking should be equally accessible to all segments of the population [1]. In this paper we elaborate a multidimensional framework for equitable computer science instruction that aims to disrupt inequities that occur at the level of classroom interaction.

Equity can be conceptualized in terms of students' access to material resources (e.g., computer labs, substantive coursework beyond basic computer literacy) [12]. We argue that equitable classrooms also require pedagogical practices that account for the social and emotional aspects of learning [1][4]. To that end, we propose a framework for equitable teaching that consists of four distinct but interrelated dimensions: (1) access to rich course content; (2) access to quality instruction; (3) access to identities as computer scientists; and (4) access to productive peer relationships. To illustrate each dimension, we report on our experience designing and implementing a suite of equity-oriented teaching practices in a computer science course for students entering the sixth grade.

This article makes two primary contributions that can support the computer science education community in researching and implementing equitable pedagogy. First, it provides a theoretical framework consisting of four dimensions of equity that we believe are important for designing and evaluating teaching practices that support equity in computer science classrooms. And second, it presents our experiences developing and implementing concrete practices that may support educators in building equitable learning environments.

2. THEORIZING "EQUITY"
Educational equity is predicated on the idea that all students should have equal access to the particular resources they need to succeed in school [9]. Some of these resources, such as rich course content and quality instruction, are material in nature. In their longitudinal study of computing programs at three, racially diverse Los Angeles high schools, Margolis, Estrella, Goode, Jellison Holme, and Nao [12] illuminated some of the ways in which differences in students' access to material resources can lead to systemic inequities in computer science education. For example, while two of the schools in the study had ample technological resources (e.g., computer labs), the only coursework available to students focused on basic computer literacy—these schools were "technology rich but curriculum poor" ([12], p. 14).

And yet, Margolis and her colleagues found that disparities in material resources were but one driver of inequity. Unlike the first two schools, the third school in their study did offer advanced computing courses (i.e., rich course content). However, few African American and Latina/o students enrolled in the school's Advanced Placement (AP) Computer Science class, which was instead disproportionately dominated by White male students. In part, the researchers attributed this participation gap to a learning environment that positioned less experienced and less confident students—all of whom in this particular study happened to be either female or from historically underrepresented racial groups—as less capable of learning the subject. In their words, such an environment "constrains the choices [students] make about whether or not to engage in AP computer science specifically, and the AP pipeline in general" ([12], p. 87).

Other researchers have also investigated the ways in which the quality and configuration of computer science learning environments mediate student engagement. Garvin-Doxas and Jackson [8] analyzed undergraduate computer science classrooms that engendered an impersonal, "defensive climate," which they attributed to particular classroom norms. For instance, instructors rarely used students' names, and students rarely interacted with instructors or even spoke in class. Further, similar to the AP Computer Science classroom in the previous study, status hierarchies formed between students with prior experience and students who were relatively newer to the discipline. These hierarchies were reproduced in everyday classroom interactions, including moments when certain "high-status" students asked public questions that—from the researchers' perspectives—seemed to be more about showing off expert-level knowledge than genuine requests for clarification.

Together, these studies speak to the role of *non-material* resources in student learning. Specifically, classroom environments that engender status hierarchies between students are problematic because they can cause those students who are positioned as "low-status" to disengage from the learning process [4]. Disengagement can occur because hierarchies prevent students from forming relationships with their peers and seeing them as resources [4], but also because, as learning theorists have argued, learning and identity are closely intertwined [13][16]. That is, when students do not perceive themselves as capable of academic success, they are less likely to persist in learning: If a computer science learner does not believe that she or he can be successful in learning computer science, then why make the effort? [7] In this sense, students' access to identities as computer scientists and access to peer relationships are both components of equity, and carry at least as much weight as students' access to physical resources, rich course content, or highly qualified teachers [9].

To summarize, then, equitable teaching approaches must simultaneously account for students' access to both the material *and* non-material resources necessary for learning. The multi-dimensional framework for equitable instruction we present here is grounded in this conceptualization of equity (see Figure 1). In subsequent sections we describe the specific pedagogical practices we implemented, as well as the research base that informed their design.

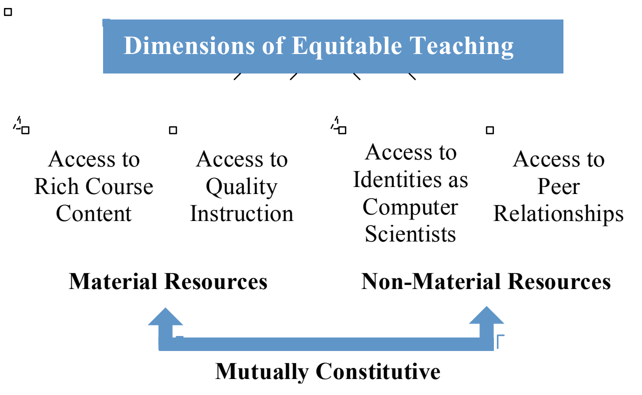
Dimensions of Equitable Teaching
Access to Rich Course Content
Access to Quality Instruction
Access to Identities as Computer Scientists
Access to Peer Relationships
Material Resources
Non-Material Resources
Mutually Constitutive

Figure 1. Framework for Equitable Teaching

3. DESCRIPTION OF CLASSROOM SETTING

Our team of seven researchers collaborated to design and implement plans for equitable instruction in a course titled, "Creating Music, Movies, and Games with Computers." The course took place over three weeks in the summer of 2012, and was part of a university-sponsored summer enrichment program. The enrichment program is designed for academically high-achieving students, and students without need-based financial aid paid 650 dollars to participate in the course.

There were two offerings of the course: a morning class and an afternoon class. Overall, there were 36 hours of instruction per course. Each course offering met four days a week for three weeks, and each class session lasted 3 hours, consisting of two 90-minute sessions separated by a 30-minute break. During the course students used the following programming languages: Scratch (24 hours), BYOB (3 hours), and Logo (1.5 hours) [14], and completed a final project in their choice of these languages (7.5 hours).

The seven researchers met daily throughout the three weeks to discuss the experiences and perceived needs of the students. During the class sessions, three of the researchers served as ethnographers, and took detailed field notes about the learning environment and students' experiences within it. The remaining four researchers worked as instructors for the course. Two of the instructors, one female and one male, facilitated the majority of whole-class discussion. One of these instructors originally designed the course and had taught three previous offerings, while the other instructor had taught the course the previous summer. The two remaining instructors were female and had never taught this course but had experience with computer science.

Forty-five students were enrolled across both classes: 22 students in the morning class, and 23 students in the afternoon class. With the exception of two students who were entering the fifth grade in the fall, all of the students were entering the sixth grade. Based on gender data collected by the enrichment program, 23 students were parent-identified as female and 22 students as male (51% female, 49% male). Although the female-male ratio was lower in the morning class (43:57 in the morning class, and 59:41 in the afternoon class), we note that these demographics do not mirror the underrepresentation of women at the college or career level for computer science [5]. And for the purpose of this paper, we will refer to our course as "gender balanced."

A typical class session involved work within an online curriculum, a 15-minute paper-based assessment, lecture with whole class discussion, and a 30-minute paper-based assignment to be completed at home. For each of the 12 class sessions, the online curriculum contained a set of formative assessments and programming tasks. All students began with the same activity each day and were allowed to move at their own pace. Topics covered included: conditionals, looping, iteration, problem decomposition, and event-based programming. Previous research provides additional details regarding the course content [10][11].

Each day the students alternated between working in the online curriculum individually and working with a partner. On "solo programming" days, although students worked on their own computers, they were still assigned a partner who they were encouraged to turn to for help. On days that students worked with other students, we employed a practice referred to as *pair programming*, in which students are paired in groups of two (and, in our class, occasionally groups of three) while working together on one computer [3]. The student who was in control of the mouse and keyboard is called the "driver," while the student responsible for guiding the driver and giving instructions or advice is called the "navigator."

Further details on the curriculum and pedagogy of the course are discussed below in relation to our specific equity-based goals.

4. IMPLEMENTING EQUITY IN PRACTICE

In this section we introduce the four dimensions of our equity framework, and exemplify each with practices implemented in our course. Our primary pedagogical goal was to build a learning environment that gave all of our students equal access to both the material resources (i.e., rich course content, quality instruction) *and* non-material-resources (i.e., identities as computer scientists, peer relationships) needed for learning. The practices described here and listed in Table 1 constitute a representative sample of our teaching approach.

Table 1. Pedagogical practices listed by dimension.

Dimension	Pedagogical Practices
Access to Rich Course Content	• Emphasizing Multiple Solutions • Using Metaphors to Introduce Concepts • Debugging by "Acting Out"
Access to Quality Instruction	• Tracking Student Progress • Customizing Teaching Plans for Individual Students • Using iClickers for Formative Assessment
Access to Identities as Computer Scientists	• Exposing Students to a Diverse Set of Computer Scientists • Managing Public Displays of Status • Encouraging Connections to "Out-of-School" Identities
Access to Peer Relationships	• Strategically Partnering Students • Encouraging and Structuring Peer Interaction • Modeling Ideal Peer Collaboration

4.1 Access to Rich Course Content

The first dimension of our framework for equitable teaching concerns students' *access to rich course content*. As was noted earlier, curriculum mediates students' opportunities to learn. That is, taking a course in basic computer literacy (e.g., word processing) does not afford the same level of learning opportunities as a course in computer programming [12]. Our course content covered the same material as the first four weeks of an introductory computer science course at a local university. Moreover, the curriculum required students to learn above-grade level mathematics content. For these reasons, we consider the course content presented to our rising sixth grade students to be rich and, for our students, challenging.

In addition to the actual content of the curriculum materials, *access to rich course content* includes teaching practices that make the curriculum accessible. We implemented three practices to help students develop robust problem solving strategies, and to help students access this rich computer science content, which is often seen as abstract and inaccessible. These practices were: 1) emphasizing multiple solutions; 2) using metaphors to introduce concepts; and 3) debugging by "acting out."

4.1.1 Emphasizing Multiple Solutions

Traditional approaches to STEM education tend to convey the message that problems have only "one right answer." This is problematic because it can discourage students from building robust problem solving repertoires and perpetuate narrow, procedural views of rich disciplines like computer science [1]. Instead, we structured our curriculum to emphasize the idea that problems have multiple solution paths. For example, the online curriculum included a task that required students to draw a rainbow in Scratch. The curriculum described two different methods for creating the rainbow, and students were afforded the opportunity to select one of these methods or invent a method of their own.

The theme of "multiple solutions" was also promoted on homework assignments. For instance, a typical homework task involved students being given both a problem and its solution script, and then asked to provide an alternate script that would solve that same problem in a different way. In general, open-ended problems were a staple of the course, and during whole-class discussions the instructors would not let students settle on an initial solution unless alternate solutions had been considered.

4.1.2 Using Metaphors to Introduce Concepts

Students with little or no prior experience in computer science can find the subject abstract and thus difficult to learn. To address this issue, instructors employed specific metaphors to introduce students to the concepts of programming. For example, on the first day of class students were told that they would be learning a language with which to communicate with the computer [6]. We did an activity in which one of the instructors acted as a "computer" and the other instructor acted as a computer scientist. The "computer" left the classroom, and the class agreed on an image (e.g., a house) for the "computer" to draw on the chalkboard, knowing that the "computer" could only understand basic commands about shape, size, and direction. When the "computer" returned, the "computer scientist" gave instructions on how to draw the image without looking at what the "computer" was drawing. The result was a nonsensical picture, as no instruction was clear enough to produce a coherent version of the desired image. This metaphorical activity, which students then repeated with their partners, was intended to teach students about the relationship between computer and computer scientist.

4.1.3 *Debugging by "Acting Out"*

In keeping with our effort to make computer science more concrete for our students, we sought to support our students in using physical intuition to solve problems [14]. One practice we encouraged students to use in debugging was to physically "act out" the code they had written. As a foundation for incorporating this strategy, students participated in an interactive, outdoor geometry exercise where students acted as "sprites," while their partners gave them instructions to trace squares, triangles, and circles that had previously been drawn in chalk on the ground. Teachers continued to engage the strategy of "acting out" when helping students, and would often have the student instruct them as they acted out the motions for the students to observe. In fact, we noticed a number of occasions when students independently used this strategy with a partner to successfully debug a program. Thus, by having students physically act out code, we aimed to provide a more concrete understanding of programming concepts that are traditionally difficult to grasp.

4.2 Access to Quality Instruction

The second dimension of our framework pertains to students' *access to quality instruction*. While the previous dimension referred to students' opportunities to engage with appropriate and rich computer science curriculum, this dimension focuses on equity-oriented pedagogical practices that are not necessarily specific to computer science. Below we note structural factors that contributed to students' *access to quality instruction* and highlight three practices in the service of this goal: 1) tracking student progress; 2) customizing teaching plans for individual students; and 3) using iClickers for formative assessment.

4.2.1 *Structural Factors*

A common problem in education is that large class sizes make it difficult for teachers to provide students the individualized support they need. Several structural factors in our particular learning setting mitigated this issue for our students. First, course enrollment in the summer enrichment program was capped at 24 students. Second, many courses in the program were co-taught by two instructors. In our case, not only did the two primary instructors have extensive experience teaching computer science to upper elementary students, but there were two additional instructors who further reduced the student-teacher ratio. These additional teacher resources allowed us to provide prolonged, individual attention to students who were experiencing difficulty, as well as to students who had been absent from a previous class session. In some cases an instructor worked with an individual student for more than an hour, scaffolding where appropriate. While many of these structural affordances are not generalizable, the following practices can be extended to classrooms with fewer resources.

4.2.2 *Tracking Student Progress*

Although some students will ask for help when they get stuck, others will struggle without calling for an instructor. This can present an inequity in access to instruction, because students who are less comfortable asking for help do not have access to the same resources.

The majority of class time was spent with students working individually or in pairs on the activities from the online curriculum. In order to identify students who were progressing more slowly, or who may need additional support to complete the activities for the day, we maintained a document tracking students' progress through the online curriculum. This document was a matrix consisting of our student roster along one axis and selected steps in the online curriculum along the other. Instructors would mark timestamps when a student was observed completing a particular activity. This tracking document allowed us to approach students that we believed to be in need of additional support, which is a practice that has been referred to as "targeted tutoring" [15]. Having four instructors in the room provided an uncommon level of resources to support students in this way, but the tracker document is a replicable practice that focuses an instructor on the needs of students that may not otherwise seek assistance.

4.2.3 *Customizing Teaching Plans for Individual Students*

Assuming that all students learn at the same pace and through the same methods can make instruction inaccessible to students progressing both slower and faster than average. Using the idea that "equity" does not necessarily mean that all students have the same needs, we attempted to address each student's unique needs instead of employing a "one size fits all" approach [9]. A primary support for this customized instruction was team conversations before and after class sessions. During these conversations, instructors and ethnographers shared concerns about particular students and developed plans to support these students. For example, one of our students had prior experience with Scratch and tended to go through the curriculum very quickly and with minor difficulties. The teachers were conscious to give the student extra challenges when appropriate. On the other hand, with a student who experienced difficulty, we attempted to identify strengths in this student's understanding and build upon that when working with this student one-on-one.

4.2.4 *Using iClickers for Formative Assessment*

To assess students' current understanding of a topic, we integrated into our daily routine a set of formative assessment questions in the form of multiple-choice questions. We used iClickers, which are wireless devices used to anonymously vote on multiple-choice questions projected to a classroom of students.

In the case of conceptually rich questions, we had students discuss their solutions with a partner. One concern we had about this arrangement was its potential to reify status hierarchies between students. That is, a student who does not know the correct answer may be embarrassed by attempting to explain what they later find out is an incorrect answer, or one student may dominate the discussion and not provide opportunities for other students to explain their reasoning. To avoid unnecessary embarrassment for students, we made a habit of publicly announcing the correct answer before students discussed their responses with each other. This provided the opportunity for students to work together to explain in their own words the answer or solution without concern that their explanation justifies the wrong answer choice. To avoid one student dominating the discussion, we directed students to alternate which partner should be the first to explain. This was intended to provide both partners equal opportunity to explain their answer even if they were not the more confident or dominant student in the partnership.

4.3 Access to Identities as Computer Scientists

The third dimension in our framework concerns students' *access to identities as computer scientists*. Our commitment to this dimension of equity stems from research demonstrating that students' self-perceptions of their disciplinary capabilities are closely related to the ways in which they take up or reject opportunities to learn [13]. Put simply, we wanted our students to identify as someone capable of doing computer science, and to accept the idea that with enough time and effort, anyone could become a computer scientist. In this section we describe three practices intended to facilitate students' access to productive domain identities: 1) exposing students to a diverse set of computer scientists; 2) managing public displays of status; and 3) encouraging connections to "out-of-school" identities.

4.3.1 Exposing Students to a Diverse Set of Computer Scientists

Part of supporting students' identities as computer scientists was to provide them an expansive sense of who could be a computer scientist, and we did this by creating a classroom that showcased diversity. The team chose 12 computer scientists—six males and six females from historically underrepresented racial and ethnic groups—and displayed prominently on a classroom wall their names, pictures, and brief quotes about why they love computer science. Each day we introduced one of these individuals, who in addition to their ethnic and gender diversity, also worked in a variety of fields including healthcare, film, and business. Beyond these computer scientists, the gender balance of both the instructional team and the students themselves may have implicitly sent the message that both genders are equally capable of being computer scientists.

4.3.2 Managing Public Displays of Status

Students' access to identities as computer scientists is also mediated by public displays of status, which can result in a defensive classroom climate [8]. Based on our previous years of collective teaching experience, we anticipated several ways in which student behavior might create status hierarchies. One way students position themselves (and others) is in whole-class discussion. Students who frequently raise their hand and answer questions correctly can gain higher status because they appear more confident or somehow innately more capable than students who may be just as proficient but do not raise their hand.

To address this issue, we used a system to randomly call on students. All students were told from the first day that when they were called on, they could "pass," and it was emphasized that "passing" should not be viewed negatively. The randomization of student contributions helped avoid discussion being dominated by a limited number of students. This method allowed us to facilitate student participation in a more equitable fashion.

Another potential problem we sought to preempt was the perception that "faster" students are "smarter" students. In the past, we had observed that when students finish an activity before the rest of the class, they tend to display this publicly, which may unintentionally be perceived by their classmates as signifying domain competence. We implemented several practices to mitigate this issue. First, instead of collecting students' paper-based assessments as students finished them, we collected them all at the same time when the instructors saw that most or all of the students were finished. If a student finished early, she or he would begin working in an "extra time" packet provided at the beginning of the course. Second, if on a given day students finished the core lessons of the online curriculum, the last activity for that day was open-ended, thereby engaging students who had moved more quickly through the curriculum. Of course, because students could look around the room and see what their peers were working on, we acknowledge that these practices by themselves could not completely prevent public displays of status. However, these practices helped us to work toward that goal.

4.3.3 Encouraging Connections to "Out-of-School" Identities

Educational researchers have argued that, in part, learning depends on the synchronization between students' personal, out-of-school identities and their in-class domain identities [2]. That is, the more students are able to bring in "who they are" into the classroom, the more likely they are to engage and persist academically. One practice we used to foster this connection was to give students autonomy to design their own final projects, which were shown on the last day of class at an open house attended by friends and family. For instance, one student was extremely fond of horseback riding, and chose to build a computer game around it. The opportunity for students to design their own final projects encouraged students to make a connection between their "personal identity" and "computer scientist identity," thereby fostering the notion that students are capable of being computer scientists.

4.4 Access to Peer Relationships

The final dimension of our framework for equitable teaching pertains to students' *access to peer relationships*. Social relationships are a central component of classroom learning because when students know and treat their peers with respect, they are more likely to view them as resources to learn from [1]. We implemented the following practices to promote all students' access in this area: 1) strategically partnering students; 2) encouraging and structuring peer interaction; and 3) modeling ideal peer collaboration.

4.4.1 Strategically Partnering Students

Students entering a new class will naturally gravitate toward classmates they previously know, and these peer relationships can be a positive influence on learning. However, this can be problematic for students who do not know anyone in the class, and therefore do not have access to an established social group. To address this potential inequity, we strategically assigned students' to new seats each day so as to maximize their opportunity to meet and work with new classmates. Taking into account students' temperament and interaction patterns over time, we also attempted to pair students who we thought would be more likely to engage in productive collaboration (i.e., without one student dominating the conversation or physical resources, without discouraging a weaker student, and without becoming distracted with off-task behavior). As a rule we avoided partnering the most advanced students with students who we perceived to be lacking confidence or to be experiencing extreme difficulty with the material.

4.4.2 Encouraging and Structuring Peer Interaction

A critical component of students' access to peer relationships was for students to come to respect their partners as mutually integral to their respective learning experiences. To facilitate familiarity between partners, we allotted several minutes each day for partners to get to know each other on a social level by discussing a light-hearted prompt, such as "would you rather be 15 feet tall or 2 inches tall?" As previously mentioned, instructors also emphasized that there were multiple ways of determining an answer, and that there was value in understanding how their partners thought through a problem. Finally, when students were doing independent work the teachers encouraged them to consult with their partners when they had a question about a particular task prior to requesting assistance from a teacher.

On pair programming days, it was necessary to provide more explicit interaction structures. For example, every five minutes a

Scratch program played music to signal the switching of roles between "driver" and "navigator." Occasionally some students would show reluctance to give up the "driver" role. To ensure students shared duties in an equitable way, we had them enact an explicit routine that involved both partners standing up, exchanging a "high five," and switching seats.

4.4.3 Modeling Ideal Peer Collaboration

A final practice we implemented to foster productive peer relationships was to have the instructors "role play" exaggerated versions of positive and negative partner interactions as a way of modeling desired student behavior. Themes discussed in the role plays included: encouraging students to ask their partners for help, respecting the switching of roles from "driver" to "navigator" during pair programming, not boasting when they got a question correct, not putting classmates down for going at a different pace, checking in with partners about coding choices, and staying focused. While students often found the teacher-led role-plays amusing, on multiple occasions we also observed students using language and specific problem solving strategies presented during the role-plays in their interactions with each other. By providing a model for students to emulate, we sought to scaffold access to positive peer relationships within the framework of the course content.

5. DISCUSSION

In this paper we outlined a multi-dimensional framework for equitable teaching in computer science. Further, we demonstrated how each dimension was generative of specific pedagogical practices aimed at equalizing students' access to the resources necessary for learning. Equity can be defined as students' access to material resources, such as curriculum and qualified teachers. Leveraging research from a wide range of fields, including psychology, sociology, and mathematics education, we argued that *non-material* resources (e.g., identities as computer scientists, peer relationships) also play a significant role in the learning process, and therefore also hold implications for equity.

For the sake of clarity, we presented each of the four dimensions in our framework as distinct. In reality, however, these dimensions are necessarily interrelated. For example, rich course content is about more than exposing students to certain skills and concepts. It is also about identity, in that it sends a message to students about their capabilities: what kinds of learners they are and what kinds of learners they can become. Conversely, if classroom status hierarchies push students to become increasingly disidentified from a content domain like computer science, they will also become less willing to engage with a rich, challenging task. In general, then, the four access points we highlight are mutually constitutive of each other, and equitable outcomes are more likely when all of the dimensions are considered in parallel.

We should also note that although prior research gives us reason to hypothesize that the practices we implemented were effective in promoting equity, our empirical study of our teaching is ongoing, and at present we do not purport to make such a claim. Still, we are hopeful that educational researchers and practitioners will find ways to adapt elements of our approach to the particular circumstances of their local teaching and learning contexts.

6. ACKNOWLEDGMENTS

We would like to thank the students and staff of the summer enrichment program. The research reported here was supported in part by a grant from the National Science Foundation (Award #1044106), the Institute of Education Sciences pre-doctoral training grant R305B090026 to the University of California, Berkeley, and a NAEd/Spencer Dissertation Fellowship. The opinions expressed are those of the authors and do not represent views of the National Science Foundation, the Institute of Education Sciences, the U.S. Department of Education, or the National Academy of Education/Spencer Foundation.

7. REFERENCES

[1] Boaler, J. (2008). Promoting 'relational equity' and high mathematics achievement through an innovative mixed-ability approach. *British Educational Research Journal, 34(2)*, 167-194.

[2] Cobb, P., & Hodge, L. L. (2002). A relational perspective on issues of cultural diversity and equity as they play out in the mathematics classroom. *Mathematical Thinking and Learning*, 4(2 & 3), 249-284.

[3] Cockburn, A. & Williams, L. (2001). The costs and benefits of pair programming. In G. Succi & M. Marchesi (Eds.), *Extreme Programming Examined* (pp. 223-243). Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

[4] Cohen, E. G., & Lotan, R. A. (1995). Producing equal-status interaction in the heterogeneous classroom. *American Educational Research Journal, 32(1)*, 99-120.

[5] Computing Research Association. Taulbee Survey, 2010-2011. http://www.cra.org/resources/taulbee/.

[6] http://csunplugged.org "Marching Orders: Programming Languages."

[7] Dweck, C. S. (2007). Mindset: The new psychology of success. New York, NY: Random House, Inc.

[8] Garvin-Doxas, K. & Barker, L. J. (2004). Communication in computer science classrooms: Understanding defensive climates as a means of creating supportive behaviors. *Journal of Educational Research in Computing, 4(1)*, 1-18.

[9] Gutierrez, R. (2007). Context matters: Equity, success, and the future of mathematics education. Paper presented at the Proceedings of the 29th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education.

[10] Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education. 21(2).* 105-134.

[11] Lewis, C. M. & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. *ACM SIGCSE Bulletin. 43(1).* 57-62.

[12] Margolis, M., Estrella, R., Goode, J., Jellison Holme, J., & Nao, K. (2008). Stuck in the shallow end: Education, race, and computing. Cambridge, MA: MIT Press.

[13] Nasir, N. S., & Cooks, J. (2009). Becoming a hurdler: How learning settings afford identities. *Anthropology & Education Quarterly, 40(1)*, 41-61.

[14] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books, Inc.

[15] Titterton, N., Lewis, C. M., & Clancy, M. (2010). Experiences with lab-centric instruction. *Computer Science Education (Ed. Y. Ben-David Kolikant) 20(2)*, 79-102.

[16] Wortham, S. (2006). Learning identity: The joint emergence of social identification and academic learning. New York, NY: Cambridge University Press.