Defining and Designing Computer Science Education in a K12 Public School District

Chris Proctor*
Stanford University
Stanford, CA
cproctor@stanford.edu

Maxwell Bigman†
Harvard University
Cambridge, MA
mbigman@gse.harvard.edu

Paulo Blikstein
Columbia University
New York, NY
paulob@tc.columbia.edu

ABSTRACT

Computer science is poised to become a core discipline in K12 education, however there are unresolved tensions between the definitions and purposes of computer science and public education. This study's goal is to explore how logistical and conceptual challenges emerge while designing a comprehensive K12 computer science program in a public school district. While the policy infrastructure for K12 computer science education is rapidly developing, few districts have yet implemented computer science as a core discipline in their K12 programs and very little research has explored the challenges involved in putting ideas into practice. This study reports on a committee designing a comprehensive K12 computer science education program at a small public school district in California. Through a grounded-theory qualitative interpretation of committee-member interviews and board meeting transcripts, we surfaced three themes which were the primary points of tension: how computer science is defined, how it ought to be taught, and what process ought to be used to answer these questions. Grounding these tensions in the academic discourse on K12 computer science education, this study offers recommendations to other districts designing comprehensive computer science education and suggests future directions of computer science education research that will be most useful to stakeholders of these processes.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **Computing literacy**; **K-12 education**; *Computational thinking*;

ACM Reference Format:
Chris Proctor, Maxwell Bigman, and Paulo Blikstein. 2019. Defining and Designing Computer Science Education in a K12 Public School District. 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, Article 4, 7 pages. https://doi.org/10.1145/3287324.3287440

*Equal first author
†Equal first author

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02…$15.00
https://doi.org/10.1145/3287324.3287440

1 INTRODUCTION

Even though computers shape our political, economic, and social lives, K12 education in the United States does not include comprehensive computer science. When they are available at all, opportunities to learn computer science are marginal and piecemeal. AP CS enrolls mostly high-performing, college-bound students. Some students learn programming in extracurricular activities like robotics and maker spaces and clubs. Equity-focused interventions such as Exploring Computer Science [16] and Hour of Code have made progress in exposing more students to computer science. A thriving economy of educational technologies and out-of-school learning opportunities has emerged for those who can afford them. Computation rivals textual literacy in societal significance, but our schools do not systematically prepare children to participate as workers or citizens.

The computer science education community has made substantial progress on developing the infrastructure needed for comprehensive K12 CS education. Several CS curriculum standards have been published [13, 27]. The K-12 CS Framework [1] united the major stakeholders to produce a document intended as guidance to those developing standards and curricula. Various states have adopted existing computer science standards while others, including California and New York, are developing their own. At the same time, states are developing teacher credentialing standards [18] and teacher preparation programs to train computer science teachers.

As this infrastructure is still emerging in the United States, there are few examples of K12 CS program implementations, and very little research has been published on them. Several large urban school districts, including New York [29], San Francisco [28], and Los Angeles [19], are in the process of implementing K12 CS. The initial evaluation report on New York's CS4All initiative emphasizes the need for schools to develop local visions for computer science [29], and calls for case studies of school-based implementations (p. 55). We have been unable to find case studies documenting the process of designing and implementing K12 CS at smaller school districts. As other school districts consider making computer science a core K12 subject area, reports of prior experience will be valuable.

This case study of a small suburban school district's three-year process of designing a K12 CS program addresses this gap in research. Our analysis of interviews with committee members and presentations to the board of education surfaced three themes which were the focus of discussion and conflict in the committee and the broader community:

(1) What is the district's vision for computer science?
(2) How should the district's K12 CS program be implemented and taught?

(3) What process should be used to answer these questions (and who should participate)?

This study characterizes the development of each theme, grounds each in the broader discourse of computer science education, and offers recommendations for how other communities can engage productively with them. This study may be of interest to stakeholders of other school districts designing computer science programs, as well as academics studying teaching and learning in K12 CS. Stakeholders in other school districts–whether they are designing district-wide computer science programs or local implementations within large districts–may recognize the themes reported in this case study, and be able to avoid some of the conflict encountered in this district's process. The committee which is the focus of this study relied on existing research, but found it necessary to build their own shared understanding of what computer science is and how it should be taught. For researchers studying teaching and learning in computer science, the opportunity to see how academic discourse is taken up by practitioners may inform future research.

2 BACKGROUND

There is a substantial body of work considering teaching and learning in K12 CS. Although this study uses a grounded-theory approach [10] to surface the ways in which committee members came to understand computer science, this section summarizes the research which was the context for the committee's work and which frames the conversation to which this study contributes. Barr & Stephenson [3] argue for several ways the computer science education research community can contribute to the design and implementation of K12 programs. These include each of the major themes, listed above, which emerged from the interviews with committee members and board meeting transcripts.

Barr & Stephenson [3] emphasize the importance of uniting behind an "operational definition" (p. 49) of what is to be taught[1]. Each of the computer science standards documents has given a definition of computer science, and has acknowledged the difficulty of defining a young and dynamic field. Strategies for defining computer science fall into three broad groups. The first is identifying the fundamental ideas [8] or essential questions [30] at the heart of a field. This approach has been particularly attractive for computer science, which can be seen as a way of working directly with powerful ideas and thinking about thinking [22]. A second approach considers a field not as an abstract body of knowledge, but as a set of practices enacted within a community. To borrow a phrase from Roy Pea, computer science is what computer scientists do. An attractive feature of this framing is that it acknowledges cognition as situated [7] in the context of particular identities, cultures, tools, and spaces which can be important factors in equitable participation [4, 20, 21]. The K-12 CS Framework [1] and the revised CSTA standards [13] each define computer science in terms of concepts and practices. A third, more pragmatic, approach defines a field in terms of competencies. What exactly can experts do? Denning [12], for example, argues that computational thinking is a poorly-defined construct responsible for much confusion in K12 CS education, and that we would do better to focus on clearly-measurable outcomes aligned with the kinds of tasks computer scientists do in their daily work.

In considering how a computer science program ought to be implemented and taught, the computer science education community has sometimes emphasized interaction with computers over pedagogy. This is understandable, as thinking with computers is a core part of computer science [2], and many computer scientists have had profound personal experiences doing so. Computer science education pioneers such as Papert [22] valorized programming as a highly generative learning space, and software such as Scratch [24] has introduced millions of children to programming. Learning directly from computers remains an appealing option for scaling computer science education: Hour of Code is structured as student-guided tutorials with minimal peer or teacher interaction with no teacher expertise required [11]. However, research on supporting the development of interest and learners' identities in the face of negative stereotypes has shown the importance of supportive communities and personal relationships [4, 20, 21]. This research has spurred development of courses such as Exploring Computer Science [16], which revealed some of the challenges and opportunities involved with implementing computer science courses in a high school setting.

While the computer science education community has advocated for both top-down and bottom-up approaches, current K12 CS implementation work largely addresses policymakers and envisions centralized expert design being disseminated to schools and teachers for implementation via standards, teacher credentialing, and professional development. This may be driven by the ease with which digital resources can be distributed and by the corporate decision-making structures which have dominated education for much of the past century [9]. The alternative, distributed participatory design rooted in the values and identity of particular school communities, will become more feasible once there is a larger corps of K12 CS teachers to help lead the process. One of the findings of this study is the limits of scaling centralized decision-making: local stakeholders may need to construct shared understandings of fundamental questions about computer science.

3 METHODS

This study was conducted at a small wealthy suburban school district in California in which a committee composed of students, parents, teachers, and school leaders has spent the last three years designing a K12 CS program. The committee also generated implementation proposals, addressing questions such as how computer science should be integrated into elementary, middle, and high school, whether computer science should be a stand-alone course or integrated into other subject areas, whether there should be a computer science department, and whether a computer science course should be a graduation requirement. The district is unusual in its level of affluence and social capital. The high schools already offer numerous computer science courses, and several of the committee members have participated in K12 CS education leadership in various capacities.

[1]Barr & Stephenson address computer science but focus on computational thinking. As we understand the article, computational thinking is suggested as a candidate for an operational definition of K12 computer science.

Table 1: Frequency of qualitative codes in committee member interviews and board meeting transcripts

Code	Count
Theme 1: Definition	392
competencies	29
concepts	29
defining_education	46
framings	49
importance_of_defining_cs	8
practices	71
rationales	119
Theme 2: Curriculum and instruction	578
curriculum	394
implementation	45
pedagogy	139
Theme 3: Process	486
authorities	92
committee_participation	49
committee_roles	33
cs_inclusion	33
decision_making_structure	24
district_leadership	24
identity	199
politics	31

The texts analyzed in this study include semi-structured interviews with 9 core committee members and transcripts from presentations and public discussions at the district's Board of Education meetings in May 2017 and May 2018. The interviewees included parents, teachers, and school and district leaders. We used the committee's biweekly meeting minutes, written correspondence, and process documents to support our analysis.

We used a grounded theory approach, allowing themes to emerge from an iterative process of open coding. Because our goal was to understand how participants understood and framed the issues, our coding was primarily in vivo [25], noting how language is used in individual and social meaning-making, and in persuasive rhetoric. As we organized codes into thematic categories, we wrote analytical memos documenting our thinking and connecting the emergent themes to existing bodies of research [17]. The findings we present in the next section are supported by analysis of quotations and by patterns of codes in the transcripts. In subsequent work, we plan to deepen this analysis by discussing our findings with stakeholders and outside experts.

4 RESULTS

Table 1 shows the top two levels of the tree of qualitative codes which resulted from our iterative process of coding, analytical writing, and categorizing codes. The distribution of codes provides an overview of what committee member and participants in Board of Education meetings discussed, and is the basis of our analysis. The rest of this section analyzes each of the three top-level themes in turn.

4.1 What is computer science?

The first major theme was the committee's attempt to define computer science. The committee members unanimously agreed that computer science is important (though this was contested by some students and teachers at Board of Education meetings), but they invoked different rationales for why it is important. The most common rationales given were equity (52), early exposure for later interest development (21), economic opportunities (21), and a sense that computational literacy has become a skill necessary for everyday life or digital citizenship (17). Equity was stressed by the Board of Education and public commenters (37), and less frequently by the committee members (15). These rationales for the importance of computer science closely match those articulated by experts in the computer science education community[6].

The members had fundamental disagreements about what constitutes computer science and felt that their inability to agree on one definition blocked their effectiveness. One member, who has done graduate work in computer science, worked in the field, and participated in a nation-wide computer science standards committee, said, "everyone at some level was ignorant about what computer science is. So it's hard for us to define it. It's hard for us to know what it is. Only after being on the committee was I able to refine what I consider to be a pretty reasonable definition." When asked for individual definitions of computer science, committee members often referred to computing practices such as programming, computational thinking, and collaboration (100) and less frequently to core computing concepts such as abstraction, decomposition, and hardware (19). They also positioned computer science as part of broader concepts, defining it as part of what it means to be a well-educated adult today (30) and framing computer science metaphorically (47), for example as a language or as a way of using tools.

One particularly pronounced fault line was between committee members focused on bringing more students into computer science and those focused on developing students to their full potential. There need not be any intrinsic contradiction between inclusion and rigor, but differences in how each camp construed computer science made it difficult for them to understand and support each others' priorities. Those committee members who focused on inclusion overwhelmingly described computer science in terms of practices and developing the conditions under which beginners would be willing and able to participate. One teacher, describing a successful computer science lesson, reported, "every single child was not only super engaged, but it was the creative aspect, the problem solving and the teamwork that went with it and all of these other skills that I noticed. And that made learning more fun, of course. They were learning a lot." The committee members emphasizing rigor talked in terms of competencies such as programming and solving mathematical problems using the big ideas of computer science. A teacher taking this perspective argued that de-emphasizing programming (for example, through "unplugged" activities [5]) is a disservice to students: "If you're teaching origami in computer science class and some guy out in [a wealthy city] is teaching kids how to actually write programs, I've got very little doubt which kid is likely to be in better shape coming out of those two classes in terms of being able to function well at the next level. I think it's creating a chasm

between the haves and the have-nots." This disagreement is rooted in different conceptions of what it means to do computer science.

This epistemological tension is paralleled in the academic debate over computational thinking [31], even though the term was used only 12 times in our corpus. Over the last decade, proponents of computational thinking have argued for a practice-focused vision of computer science, often including collaboration, creativity, and the construction of identities and communities where computational practices can emerge. Computational thinking has been useful to those arguing for teaching computer science at the elementary level, for interdisciplinary computer science, and as a lens on inclusion and equity. Others, likely in agreement with the teacher quoted in the previous paragraph, feel that computer science, and a useful definition of computational thinking, ought to be more narrowly defined, focusing on the application of core computational ideas through programming [12, 23]. Too heavy a reliance on computer science practices risks obscuring the powerful ideas that make the field transformative, empowering, and worth teaching.

The K12 CS framework takes an inclusive position, describing computer science in terms of concepts and practices. Four of the seven practices are identified as computational thinking, and they are privileged as "the heart of the computer science practices" (p. 67) While participants reported having read and discussed the standards documents cited above, they seldom referred to them while defining computer science. It might have benefitted the committee to consider the importance of practices and concepts in turn, as presented in the K12 framework. However, documents such as the K12 framework might also be more helpful if they voiced the epistemological tensions within the field, rather than attempting to resolve them in a unified front. Divergent definitions of computer science were an important source of tension of the committee, but they are not the only explanation for the divide described above. There were also interpersonal tensions rooted in how committee members construct identities as computer scientists and position themselves as experts. We return to issues of identity and power in the third major theme.

4.2 How should computer science be taught?

The second major theme that emerged from our research centered on the question of how computer science should be taught. The committee agreed that computational thinking should be an interdisciplinary strand woven into the curriculum in elementary and middle school, with specialists supporting classroom teachers. At the elementary level, the committee proposed incorporating unplugged activities [5] and robots into interdisciplinary lessons due to concerns about limiting students' screen time. At the middle school level, the committee proposed replacing discrete units in the math and science curriculum with computational approaches. The committee was overwhelmingly in favor of making one semester of computer science a graduation requirement, with 69% of the members supporting it, and only 15% against the idea (the other members abstained or tried to find a third way). One veteran computer science teacher cited how "every year we are losing more 700 students to computer science illiteracy and if we could just get a basic computer science class going for everyone...that hemorrhaging of knowledge in some sense could be evaded." Another computer science teacher said, "I'm definitely on the side of a requirement, if every student is forced into it, it may not have the effect it had on me, where you just throw your whole self into it and get excited about it, but at least it presents the opportunity." While there was broad support for a graduation requirement, there was substantial disagreement about what sort of course should be required. Mirroring the two different understandings of computer science, there was a group emphasizing rigor, and another group focused on exposure. The former group was interested in the AP Computer Science Principles course, citing "the number of people who have worked hard on that, including people from the university level," while the latter group was interested in emphasizing the broad applications of computer science in an inclusive manner. The committee ultimately did not reach an agreement on the nature of the required course, as they focused their efforts on first securing the graduation requirement.

The committee's proposal offered two potential paths to add the computer science requirement: by replacing another graduation requirement, or by adding a new requirement and thereby reducing the number of electives available to students. At the committee's presentation to the Board of Education in 2018, they received strong pushback from teachers from other departments, students, and some of the Board members. Two members of the History-Social Studies department presented at the Board meeting. They emphasized the importance of their courses, expressed frustration at not hearing about the committee's work earlier and questioned why every student needed computer science if they did not "desire to pursue technologies and system development." Given the nature of high school graduation requirements, adding any new requirements (computer science or otherwise) inherently is a zero-sum game. Goode & Margolis [16] came to a similar conclusion when trying to integrate their Exploring Computer science curriculum: "students' course schedules are so impacted by graduation and college-preparatory requirements that they are often not able to take any non college-preparatory electives." Debates about curricular requirements were often oriented toward the state's requirements and a possible future computer science requirement.

The students who spoke against the requirement cited the the overwhelming demands of the existing requirements, and the limited choice in their schedules. One student summed up a survey of his peers: "they're less against the idea of teaching computer science than they are with the idea of more requirements," instead suggesting that the district might "loosen up our curriculum and rethink that idea of what students must know, because right now we're all working under a system that's collapsing under its own weight." The second question of student autonomy and choice suggests a much bigger question in education: should students be able to choose their courses, or should districts determine what courses students need to take? On the other hand, one Board member argued for a requirement from an equity perspective: "if we believe that computer science is a critical skill for being a citizen in the modern world, then... we really can't accept a situation in which our female students and the underrepresented minority students are not participating in that at the same levels as other students. That really should be unacceptable to us, and I think a graduation requirement is a way of addressing that." Indeed, other implementation efforts have concluded that access is not enough to broaden participation

[15]. Furthermore, they suggest that an interdisciplnary approach to computer science education will not address the need to develop equitable practices. In combination with inclusive content and pedagogy which actively confronts negative stereotypes, widespread participation may ultimately depend on requiring students to take at least an introductory course [14, 15].

The debates about the nature of the graduation requirement suggests that it is important to include a wide variety of key stakeholders and incorporate their voices. Navigating the complexities of implementing computer science is inevitably situated in a political environment that goes beyond the discipline itself, and forces a district to consider major questions about requirements and goals. Behind these more immediate issues is the reality that the basic structure of secondary education has not changed in over a century in the US, and may be due for an overhaul. Our society's shift to computation as a medium of literacy would be well-suited as the basis of such a pivot, but waiting on such an overhaul before introducing computer science would not be an expedient strategy.

4.3 What process should be used to answer these questions?

The third major theme which emerged from our analysis was the question of how the design process ought to be conducted and who ought to participate. During interviews, committee members enthusiastically described how their identities as computer scientists developed. Their understandings of computer science were deeply grounded in these histories. The committee members who were most outspoken advocates for equitable participation had past experience with exclusionary stereotypes related to computer science and the factors which can support equitable participation. These included discussions of gender (31), race (7), social class (6), affect toward computer science (21), mentorship (4), and identities such as nerds (9). One teacher, who has decades of experience teaching math and computer science, described her initial interest in computer science: "It was just what you would expect. It was nerdy kids and nerdy me and very playful, very advanced math kind of kids and I kept wondering why we weren't getting more girls because I thought it was fun and I was a girl." She acknowledges the reality of disproportionately male participation and the stereotypes reinforcing this, while also offering herself as a counterexample to these stereotypes. From this teacher's perspective, the challenge is to let girls be nerdy too, or to let computer science be defined as playful, something other than just nerdy. No participants disagreed that culture and identity are important, but some did not describe their own development in these terms. Instead, their accounts emphasize the schools where they studied, the roles they occupied, and frequently hands-on opportunities with computers. Their equity-oriented work focused on outreach, trying to get underrepresented groups to enroll in the rigorous computer science courses, rather than redesigning the courses. To the extent that these teachers use their own histories as a guide, the link between their histories and the way they understand computer science is unsurprising.

When committee members positioned themselves as experts on computer science, they often referred to their professional or academic experience. However, these experiences did not automatically confer social capital. Some committee members presumed upon their expertise to make their claims authoritative. Others, despite years of professional experience and academic background, did not use their experience to make their voices more powerful. For example, one teacher and committee member said, "I worked for 15 years in the tech industry before I decided to teach elementary schools... My background is engineering but not computer science, though the work I was doing was programming. And so after fifteen years in the tech industry, I lost my job, I went back to school, I did my multiple subject credential, and I started teaching as an elementary school teacher." Throughout her interview, she supported her perspective on how computer science ought to be taught with detailed accounts of students' classroom experience, but avoided making general authoritative statements about what is best.

In some cases, the construction of expertise was explicitly about power and whose voice counts. One male teacher explicitly excluded the expertise of the women quoted in the previous paragraphs: "It's tough because, you know, I felt like [other male teacher] and I were the two subject matter experts in the room other than a couple of parents who were in the business. And I think both [other male teacher] and I at different times during the past couple years have, you know, threw our arms up." This teacher has strong views on what constitutes computer science and on how it should be taught, and frequently saw himself in the minority on the committee. These interactions portray computer science knowledge as situated in particular identities, contexts of use, and power relationships. This is a not a dominant perspective in computer science. Abstraction, one of the field's core ideas, relies on the objective representations of concepts[2]. As the computer science education community tackles the challenge of making computer science broadly accessible, we may also come to value a plurality of different manifestations of what it means to do computer science, each rooted in its own particular context of use.

In considering the question of process, the committee considered it a foregone conclusion that this needed to be inclusive, building a broadly-shared understanding of what computer science is, a consensus that it should be a core part of the curriculum, and agreement on how it should be implemented. The Board of Education charged the committee with including stakeholders including teachers, parents, students, and community members. At the two Board of Education presentations there was robust participation from members of all these constituencies, including committee members and those who did not participate. When, at the latest Board meeting, the committee encountered political opposition which stalled its progress, several committee members reflected on whether a different process might have been more effective. Some committee members felt that inclusion had come at the cost of efficiency (9), and felt that it would be better to move forward with a smaller, more united committee which could push for a single vision. Others argued for the opposite, that the committee should focus on community education (18) so that more teachers, parents, and students could come to an understanding of what computer science is and why it ought to be a district priority. One participant emphasized the political importance of community support: "In order to

[2]That said, some subfields, such as human-computer interaction and natural language processing, are fundamentally concerned with representing a plurality of interpretive processes.

sell this, we're going to need to have a district-wide community committee so that the community knows more about it and we get them on board." Several committee members pointed out the recursive challenge of sufficiently educating the school community about computer science so that the community would be able to meaningfully participate in designing a computer science education program.

5 DISCUSSION

Shortly before this study's publication, the district's Board of Education decided to delay enacting the committee's recommendations for a comprehensive K12 CS program with a one-semester graduation requirement. Instead, the board will convene a smaller, more focused task force for another year of study. While this committee's years of committed work undoubtedly advanced the cause of comprehensive K12 CS for every student in the district, their goal has not yet been met.

How might this experience contribute to the larger discourse of K12 CS education? The financial, cultural, and social capital which enabled this district to contemplate adopting K12 CS also set it apart from the vast majority of other districts in the United States and may limit the applicability of lessons learned here to other contexts. Would the process be less contentious in school districts which have less existing computer science expertise and fewer established programs? Would the debate be obviated in a large school district or statewide initiative, where policy can be unified and implementation guidance can be provided to individual schools? Is this a case of a district that went out too far ahead, before supporting research and policy infrastructure was in place?

We believe not. Interventions at scale, particularly those which are distributed online, may succeed in introducing students to computer science without requiring school communities to engage in the hard work of defining and designing local instantiations of computer science education. But lasting teaching and learning, which builds identities and expertise, takes place within communities of practice, populated by people acting from their own situated understandings and identities. In such an environment the issues identified in this study are likely to emerge. Participants are likely to act from their own definitions of computer science, which may be differently grounded in concepts, practices, or competencies. They are likely to be divided by competition for the scarce resources of curricular requirements, and by different opinions on whether computer science should be a standalone course or be integrated across disciplines. And all of these are likely to be shaped by participants' life histories with computer science, and the identities they have developed.

For stakeholders of schools designing comprehensive computer science programs, this study offers several lessons. It is important to realize that there are substantial, unresolved disagreements about what constitutes computer science, and that any implementation will need to take a stance on what matters most. Peoples' understanding of computer science will be situated in their own experience, and the process of developing a broadly-shared operative definition of computer science should include time to discuss these different backgrounds. In order to include participants without existing computer science experience, it may be necessary to invest in community education efforts. Despite these efforts, some people will be skeptical that computer science ought to be a priority; if they are not also included in the process they may solidify into an oppositional faction. The question of whether to require a separate computer science course may be a crux point, essential for equitable participation but also in competition with other priorities for the limited number of courses that can be required. The same factors which have historically excluded students from participating in computer science are likely to affect potential participants in the process of designing a computer science program.

For computer science education researchers, this study offers an example of how ideas from research were understood and used by one set of practitioners, and also surfaces some ways in which K12 CS education research will differ from research in the university setting. First, the divisions within academic computer science are not just an internal family affair. There may be value in further exploring the tension between different epistemological stances (concepts, practices, competencies), and in characterizing these tensions rather than trying to resolve them. It would be valuable to develop constructs which view computer science knowledge and expertise as situated in particular identities and cultures, and as part of broader educational goals. Pedagogical content knowledge [26], knowing not just the content, but also the many ways in which it can be effectively learned, is particularly important in K12 education, a setting in which success is demanded for every student and learning is more deeply intertwined with developmental processes. Such research will likely require participatory relationships between teachers and researchers. Ubiquitous issues of power and positionality between K12 teachers and academic researchers are likely to be particularly exacerbated in computer science, a field where experts (or even novices) have economic opportunities which substantially outstrip a teacher's salary.

6 CONCLUSION

Making computer science a core part of K12 education presents fundamental challenges to both fields, but we see them as productive tensions that may push each to grow and improve. Computer science developed as an unconventional, idiosyncratic field distinguished by a characteristic way of thinking, and by identities (geek, nerd, hacker) which have historically nurtured some marginalized people while excluding many others. However K-12 education is charged with designing learning experiences which work for all students, working within a system shaped by ongoing power struggles including race, gender, and social class. To meet the demands of K12 education, computer science must adapt to pluralistic ways of thinking and meet the needs of a more diverse set of practitioners. On the other hand, many have observed that our K12 public schools still fundamentally operate according to a model developed in the nineteenth century. Computer science may bring new technologies, pedagogies, ways of working, and levers for change such as transparency and analysis. This study offers an early sketch of how these tensions and possibilities may unfold.

REFERENCES

[1] 2016. K-12 Comptuer Science Framework.
[2] A. V. Aho. 2012. Computation and Computational Thinking. 55, 7 (2012), 832–835. https://doi.org/10.1093/comjnl/bxs074

[3] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? 2, 1 (2011), 48–54.

[4] Brigid Barron. 2004. Learning ecologies for technological fluency: Gender and experience differences. 31, 1 (2004), 1–36.

[5] T Bell, I Witten, and M Felllows. [n. d.]. Computer Science Unplugged. www.csunplugged.org

[6] Paulo Blikstein. 2018. Pre-College Computer Science Education: A Survey of the Field.

[7] John Seely Brown, Allan Collins, and Paul Duguid. 1989. Situated Cognition and the Culture of Learning. 18, 1 (1989), 32. https://doi.org/10.2307/1176008

[8] Jerome S Bruner. 1960. *The process of education*. Harvard University Press.

[9] Raymond E Callahan. 1964. *Education and the cult of efficiency*. University of Chicago Press.

[10] Kathy Charmaz. 1996. Grounded Theory. In *Rethinking methods in psychology*, Jonathan A. Smith, Rom HarrÃĭ, and Luk Van Langenhove (Eds.). Sage.

[11] code.org. 2018. How to teach one Hour of Code with your class. https://hourofcode.com/ac/how-to

[12] Peter J. Denning. 2017. Remaining trouble spots with computational thinking. 60, 6 (2017), 33–39. https://doi.org/10.1145/2998438

[13] CSTA Standards Task Force. 2017. Computer science standards.

[14] Joanna Goode. 2008. Increasing Diversity in K-12 computer science: Strategies from the field. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 362–366.

[15] Joanna Goode, Gail Chapman, and Jane Margolis. 2012. Beyond curriculum: the exploring computer science program. *ACM Inroads* 3, 2 (2012), 47–53.

[16] Joanna Goode and Jane Margolis. 2011. Exploring Computer Science: A Case Study of School Reform. 11, 2 (2011), 1–16. https://doi.org/10.1145/1993069.1993076

[17] Egon G Guba. 1981. Criteria for assessing the trustworthiness of naturalistic inquiries. *Ectj* 29, 2 (1981), 75.

[18] ISTE. [n. d.]. ISTE Standards for Computer Science Educators. http://www.iste.org/docs/pdfs/20-14_ISTE_Standards-CSE_PDF.pdf

[19] LAUSD. 2014. L.A. Unified Announces Larger Focus on Computer Science for K-12. (2014).

[20] Jane Margolis. 2010. *Stuck in the shallow end: Education, race, and computing*. MIT Press.

[21] Jane Margolis and Allan Fisher. 2003. *Unlocking the clubhouse: Women in computing*. MIT press.

[22] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[23] Chris Proctor and Paulo Blikstein. 2018. How Broad is Computational Thinking? A Longitudinal Study of Practices Shaping Learning in Computer Science. (2018), 8.

[24] Mitchel Resnick, John Maloney, AndrÃĺs Monroy-HernÃąndez, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and others. 2009. Scratch: programming for all. 52, 11 (2009), 60–67.

[25] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.

[26] Lee S Shulman. 1986. Those Who Understand: Knowledge Growth in Teaching. (1986), 11.

[27] Allen Tucker, Fadi Deek, Jill Jones, Dennis McCowan, Chris Stephenson, and Anita Verno. 2003. A Model Curriculum for K-12 Computer Science. (2003), 60.

[28] Bryan Twarek and Jim Ryan. 2015. The Need for Computer Science Education for All Students from PreâĂŘKindergarten to 12th Grade. https://drive.google.com/file/d/0B0TlX1G3mywqRGJKNkN2RXlZTWs/view?usp=embed_facebook

[29] Adriana Villavicencio, Cheri Fancsali, Wendy Martin, June Mark, and Rachel Cole. 2018. An Early Look at Teacher Training Opportunities and the Landscape of CS Implementation in Schools. (2018).

[30] Grant Wiggins and Jay McTighe. 2005. *Understanding by design*. Ascd.

[31] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.