Elementary Students' Understanding of CS Terms

JESSICA VANDENBERG, Educational Psychology, North Carolina State University, Raleigh, North Carolina, USA
JENNIFER TSAN, Computer Science, North Carolina State University, Raleigh, North Carolina, USA
DANIELLE BOULDEN, Learning Design and Technology, North Carolina State University, Raleigh, North Carolina, USA
ZARIFA ZAKARIA, Educational Psychology, North Carolina State University, Raleigh, North Carolina, USA
COLLIN LYNCH, Computer Science, North Carolina State University, Raleigh, North Carolina, USA
KRISTY ELIZABETH BOYER, Department of Computer & Information Science & Engineering, University of Florida, Gainesville, Florida, USA
ERIC WIEBE, Science, Technology, Engineering, and Mathematics Education, North Carolina State University, Raleigh, North Carolina, USA

The language and concepts used by curriculum designers are not always interpreted by children as designers intended. This can be problematic when researchers use self-reported survey instruments in concert with curricula, which often rely on the implicit belief that students' understanding aligns with their own. We report on our refinement of a validated survey to measure upper elementary students' attitudes and perspectives about computer science (CS), using an iterative, design-based research approach informed by educational and psychological cognitive interview processes. We interviewed six groups of students over three iterations of the instrument on their understanding of CS concepts and attitudes toward coding. Our findings indicated that students could not explain the terms *computer programs* nor *computer science* as expected. Furthermore, they struggled to understand how coding may support their learning in other domains. These results may guide the development of appropriate CS-related survey instruments and curricular materials for K–6 students.

CCS Concepts: • **Social and professional topics → Professional topics**; **Computing education**; **K-12 education**; **Student assessment**;

This material is based upon work supported by the National Science Foundation under Grant No. DRL-1721160. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Authors' addresses: J. Vandenberg and Z. Zakaria, Educational Psychology, North Carolina State University, Friday Institute for Educational Innovation, 1890 Main Campus Drive, Raleigh, North Carolina, 27606, USA; emails: {jvanden2, zzakari}@ncsu.edu; J. Tsan and C. Lynch, Computer Science, North Carolina State University, Engineering Building II, 890 Oval Drive, Raleigh, North Carolina, 27606, USA; emails: {jtsan, clynch}@ncsu.edu; D. Boulden, Learning Design and Technology, North Carolina State University, Friday Institute for Educational Innovation, 1890 Main Campus Drive, Raleigh, North Carolina, 27606, USA; email: dmboulde@ncsu.edu; K. E. Boyer, Department of Computer and Information Science and Engineering, University of Florida, 432 Newell Drive, Gainesville, Florida, 32611, USA; email: keboyer@uf.edu; E. Wiebe, Science, Technology, Engineering, and Mathematics Education, North Carolina State University, Friday Institute for Educational Innovation, 1890 Main Campus Drive, Raleigh, North Carolina, 27606, USA; email: wiebe@ncsu.edu.

1946-6226/2020/06-ART17 $15.00
https://doi.org/10.1145/3386364

Additional Key Words and Phrases: Cognitive interviewing, elementary, computer science, instrument development

ACM Reference format:
Jessica Vandenberg, Jennifer Tsan, Danielle Boulden, Zarifa Zakaria, Collin Lynch, Kristy Elizabeth Boyer, and Eric Wiebe. 2020. Elementary Students' Understanding of CS Terms. *ACM Trans. Comput. Educ.* 20, 3, Article 17 (June 2020), 19 pages.
https://doi.org/10.1145/3386364

1 INTRODUCTION

Researchers have found that negative stereotypes about computer science influence students' decisions to pursue or abandon a degree in the field [13, 25]. Holding a positive attitude towards computer programming is likewise correlated with higher self-efficacy in programming [30]. Students acquire their beliefs about topics from their first-hand experiences, from direct observation, and from evaluating what others have told them [2]. These beliefs form the basis of their attitudes, which in turn impart meaning to objects or activities [38].

With initiatives such as Hour of Code[1] and communities like CS For All,[2] students are now being introduced to computer science concepts as early as elementary school. Indeed, the new K–12 Computer Science Framework specifically emphasizes the need for young students to engage in varied types of computing [3]. In light of this instructional push, researchers need to focus on understanding elementary students' beliefs, attitudes, and experiences in computer science. Nine- to eleven-year-old students are within Piaget's [31] concrete operational stage; the majority of them should have the ability to think logically and to recognize and be able to share their unique opinions of their beliefs regarding interventions such as these.

Many of the existing curricular interventions and associated survey items for elementary-age students have been developed by domain experts who often contextualize their conceptual and technical terminology at the adult level, which they in turn expect students to master. However, children are not always comfortable with these terms, nor do they understand general concepts like "programming" in the way that the curriculum designers intend. Many researchers use self-report and attitudinal survey instruments with the implicit belief that the students' understanding of the terms and concepts matches their own. This mismatch may lead researchers to come to incorrect conclusions as to what students' experiences and attitudes are with regards to computer science and to the efficacy of their interventions. A lack of familiarity with key terms that anchor self-report items can lead to instability and an absence of consensus among students regarding the meaning of a word or phrase. This disconnect can then result in unpredictable shifts in a student's understanding of a survey item (gamma shift) or how they scale their response to the item (beta shift) pre- to post-intervention [12].

One way to access students' beliefs is to ask them [36]. The cognitive interviewing process, as detailed in Karabenick et al. [23], probes students on their understanding of what the item means and which answer they would select, in addition to other related probes. Cognitive interviewing is an iterative process in which findings from one phase necessitate refinement and further testing, with instrument development being an outcome.

In this article, we assess the students' attitudes toward and understanding of computer science (CS) concepts through a series of cognitive interviews. As part of this process, we qualitatively evaluated their responses to specific items, which then guided changes in the wording and number of items. Our analyses indicate that fourth- and fifth-grade students broadly understand computer

[1]https://hourofcode.com/us.
[2]https://www.csforall.org/.

science as *coding*. They are most comfortable using *coding* to describe what they do, for example, while making a game on Scratch[3] or giving a robot like Sphero^{TM} the right directions. Some students in our survey were able to make clear delineations between writing/building code and debugging, but many students struggled to connect coding to other subjects (i.e., science, mathematics, engineering) studied in school. These findings can inform the development and design of survey instruments as well as decisions on appropriate vocabulary to use to support elementary students in learning computer science.

2 RELATED WORK

2.1 Cognitive Interviewing with Youth

Our current work is influenced by prior research on interviewing survey-takers about their interpretation of survey items. Schwarz [37] notes that often adults struggle to comprehend the meaning of an item and that context also influences self-reports of attitude. Cognitive interviews to validate survey items are less common with children, despite important developmental differences in comprehension, understanding of abstract concepts, and working memory [49].

Cognitive interviewing as part of survey development increases the likelihood that self-report survey items are valid [23]. Cognitive validity refers to how well a respondent's thought processes align with what the survey designer intended. In other words, the goal is to determine to what extent the student thinks about and responds to the item as the designer intended. This is a layered, multi-stage process; students must read and interpret the item, determine the intent and keep this information in working memory, connect experiences from memory to the item's intent, read and interpret the answer choices, combine their inferred meaning with their own personal experiences, and then finally select their answer choice [23]. Cognitive interviewing allows us to probe students on their thinking during any part of that process.

Researchers interested in developing valid measures have utilized cognitive interviewing processes with children and adolescents in a range of topic areas. Woolley et al. [49] interviewed third- through fifth-grade students to validate a drug abuse prevention measure. Over the course of two phases of interviews, they found that wording of several items required modification to reflect a more concrete interpretation and that answer choices likewise needed to be more objective. Arthur et al. [5] took a similar approach with high school students. They were cognitively interviewed on substance abuse and risk-taking behaviors to develop an instrument assessing risk and protective factors affecting adolescents. The team had students think aloud during the entire interview process, with final results indicating that almost 100 of the 350 possible items were unclear.

In a large European health study, the authors used cognitive interviews with children and adolescents to develop condition-specific health-related measures. These measures were nested within modules that often included multiple questionnaires focused on different aspects of the disease, the emotional implications of the illness, and surveys were intended for both the patient and the caregiver [6]. The results from this cross-national study indicated that the cognitive interview process helped inform the researchers about the relevance, coherency, and appropriateness of the content for each condition-specific health module. All of these studies highlight the need to carefully listen to survey respondents and modify wording, sometimes several times, or outright eliminate items to ensure that children and adolescents can read, understand, and answer survey items.

2.2 Affective Research in CS

Affective research in CS has sought to understand the diversity of issues impacting students' interest in CS and the field in general. This is of interest in part because students' affect influences

[3]https://scratch.mit.edu/.

their cognition and learning processes [7]. Study foci include students' feelings of belongingness within the field and the need to counteract stereotypes [25], students' inaccurate preconceptions [20] or misconceptions [19] about CS, and students' range of positive and negative experiences with CS classes [21]. A comprehensive literature review of affective computing indicated that students readily recognize their own and others' emotions [35].

One mechanism for assessing affect is self-report measures [18]. These self-reports give information about how the student perceives his or her emotions at a given time and in response to a task, event, or prompt. A dearth of existing validated affective instruments specific to CS has meant that some researchers have used self-efficacy instruments contextualized in other disciplines, such as mathematics, as a proxy for measuring the impact of CS-related interventions (e.g., Reference [34]). To address this lack of appropriate CS-specific instruments, Tsai et al. [42] recently developed a self-report instrument for measuring self-efficacy for computer programming. Although they state the instrument can be used for students older than middle school, the validation was conducted with a sample of college students. A similar validation effort was done on a self-efficacy scale in Turkish with secondary school students aged 12–14 [24].

Moving beyond just utilizing Likert-type self-report, Weintrop's [45] work with high school students in three different programming environments highlights the value of asking students their perceptions of and experiences with programming. In his work, students typed open-ended survey responses and spoken interview responses were analyzed in tandem as a way of detailing students' conceptions of programming and changes over the course of the study.

The work reviewed briefly above underscores the interest in how students perceived CS and how those perceptions affected their work in CS. Our instrument development effort focuses on a younger group of students, and although the published studies inform our work, they only provide a starting point for how we should word future survey instruments for this population.

2.3 Sociocultural Theory

Children learn and develop when external—social and cultural—activities are internalized [44]. This process is nuanced, as the social and cultural activities that surround a child are highly varied. Adults, be they parents, teachers, or members of the community, are gatekeepers of immense amounts and diverse types of information. Students arrive at school, for example, with sociocultural capital and their internalization of information is mediated by language and information [9, 33]; children come to value and find differing meaning in activities by virtue of their early experiences.

Some of those early experiences fail to equip the student with the language necessary to work effectively in today's classrooms. It becomes the work of the teacher to engage the students in the discursive process of acquiring academic language [17]. In Vygotskian terms, students' everyday concepts need to transition to academic concepts—a pedagogical process termed *metamessaging* by Forman and Larramendy-Joerns [16]. By using metamessages, teachers reword students' statements to align more appropriately with the terminology expected in the classroom. John-Steiner and Mahn [22] warn that how and what students learn in out-of-school contexts and what they are taught within school directly influences school learning; therefore, children's early exposure to information is exceedingly important. Given that CS has only recently emerged as a potential academic topic, especially at the elementary level [14], it is possible that students' perceptions of this area of study and its associated language is likely to be very uneven and highly influenced by out-of-school exposure.

Given our focus on students' self-reports of their varying interests and experiences, our research objective is to develop a survey instrument appropriate for diverse upper elementary students that measures their attitudes and perspectives on CS.

Table 1. Cognitive Interviewing Probes

Standard Interviewer Probes
"Please read item number … out loud to me."
"What does that mean?" or "What is that item asking you?"
"From these [Likert] responses, which would you pick as your answer?"
"Can you explain why you picked that answer?"

Table 2. Study 1 School-level Demographics

School	Black	White	Latinx	Other	NSLP*
Atwell	36%	29%	33%	2%	98%
Ellis	6%	80%	5%	9%	6%

*NSLP denotes students eligible for Free and Reduced Lunch.

3 DATA AND METHODS

3.1 Cognitive Interviewing Process

Our cognitive interview process followed Karabenick et al.'s [23] interview probes (see Table 1). During the interview, students were asked what the item meant, which answer they would select (from *strongly disagree* to *strongly agree* on a five-point Likert scale), why that answer made sense for them, and other relevant probes (e.g., "What is engineering?"). The interviewer was encouraged to ask other germane questions emergent from the talk that would help the student express his or her understanding of the item. The students were interviewed individually by trained graduate researchers.

This protocol was used in three separate, iteratively linked studies used to both garner a better grasp of students' understanding of key computer science terms and also develop a set of refined attitudinal survey items that displays a higher degree of stability of interpretation [gamma stability; 12] across students of this age range. Below are the findings from these three studies.

3.2 Analysis

To assess students' understanding of individual items, two coders rated the students' responses to the items and interviewer prompts. Students' responses were listed verbatim and by item on a series of spreadsheets. In this way, the coders could utilize Karabenick et al.'s [23] scoring methodology for assessing overall cognitive validity and the student responses were scored on a Likert-scale from 0 to 4, indicating the coders' assessment of the student's level of understanding of the individual item. Cognitive validity indicates the student's conceptual understanding of the item as determined by alignment in their verbal interpretation of what the item means, their explanation for why they selected the answer they did, and the compatibility of their Likert response with their explanation. The coders trained on a sample set of data, then independently completed their ratings. Finally, where there was disagreement on the cognitive validity scores of the items, the coders discussed and reached consensus. Items that students struggled to understand—as determined by the coders—were examined more closely for later modification in subsequent studies. Additional details regarding analyses completed within each study appear below.

4 STUDY 1

4.1 Participants

Our participants were 33 upper elementary students (ages 9–11) in two different schools in the southeastern United States. Table 2 presents demographic data on the schools and students that participated in our study. The school names provided are pseudonyms to preserve anonymity. Atwell School is a rural school with roughly equivalent percentages of African-American, Caucasian, and Hispanic/Latinx students. Ellis School is an urban charter school with 80% Caucasian and roughly 5% each of African-American and Hispanic/Latinx students. All interviewed students also participated in a large CS educational intervention study. In that study, the students were participants in a seven-day computer science elective that implemented a coding curriculum that the researchers designed. Students in Atwell were taught by the researchers, whereas students in Ellis were taught by their technology teacher. There were 16 students interviewed at Atwell, 8 of whom were girls; 17 students were interviewed at Ellis, 9 of whom were girls.

4.2 Methods

For our initial survey items, we modified items from a validated STEM attitudes instrument [S-STEM; 43] to create a CS version for upper elementary students. More specifically, the nine original items from the Technology and Engineering Attitudes sub-scale were used. These items covered two psychological constructs: self-efficacy and outcome expectancy [47]. For example, the original item “I like to imagine making new products” was modified to “I like to imagine making new computer programs” for Study 1. In Study 1, graduate students read all of the items to the elementary students, although the children were directed to follow along, and their responses to probes were transcribed verbatim. The questions from this study are listed in Table 3 below.

4.3 Analysis

Three members of the research team engaged in thematic analysis [10]. These members represent expertise in psychology, education, and computer science. The purpose of the thematic analysis was to determine the themes that emerged from the students’ responses; in particular, if students emphasized certain experiences or concepts or introduced an example as a way of describing their perspective. After the interviews were transcribed, initial codes were determined, which were collapsed into the themes noted below. This inductive process privileged students’ perspectives and experiences, which drove changes in survey item wording. These themes were agreed upon by consensus.

4.4 Findings

We conducted specific cognitive interviews on items 1, 8, and 9. These items were selected based upon our need to determine students’ ability to understand computer programs as used initially in Item 1, but also used in the majority of the remaining items. If the students struggled to comprehend the concept in this item, we surmised it would be problematic in other items and we resolved to consider alternatives.

Regarding Item 8, we wished to ascertain to what extent students might consider STEM-based courses as being supportive of one another. There is increased interest in developing strategies for computational thinking (CT) integration into STEM subject areas [28], and this policy interest is emerging in parallel with increased researcher interest (e.g., References [39, 46]). However, it is unclear whether students are aware of these CS/CT and STEM connections. Thus, we need questions that specifically probe for this. Moreover, we reversed the wording of the items; originally the student would have been primed to consider science or math first and then computer science second. Our concern was that students might associate needing refined skills in those subject areas to do well in computer science, so by privileging computer science, students may consider how what they are currently doing may benefit their work in traditional classroom subjects.

Item 9 was of particular interest, as it was intended to assess self-efficacy and it utilized the specific phrase computer science; we were interested to learn if elementary students understood

Table 3. Item Wording Changes

Original S-STEM Wording	Study One Wording	Final Study Three Wording
1. I like to imagine creating new products	1. I like to imagine making new computer programs	1. I would like to use coding to make something new
2. If I learn engineering, then I can improve things that people use every day	2. If I learn coding, then I can improve things that people use every day	2. If I learn coding, then I can improve things that people use every day
3. I am good at building and fixing things	3. I am good at building and fixing computer program	3. I am good at building code
4. I am interested in what makes machines work	4. I am interested in what makes computer programs work	4. I am good at fixing code
5. Designing products or structures will be important for my future work	5. Designing computer programs will be important in my future jobs	5. I am interested in what makes computer programs work
6. I am curious about how electronics work	6. I am curious about how computer programs work	6. Using code will be important in my future jobs
7. I would like to use creativity and innovation in my future work	7. I want to be creative in my future jobs	7. I want to use coding to be more creative in my future jobs
8. Knowing how to use math and science together will allow me to invent useful things	8. Knowing how to use math and science will help me to create useful computer programs	8. Knowing how to code computer programs will help me in math
9. I believe I can be successful in a career in engineering	9. I believe I can be successful in computer science and programming	9. Knowing how to code computer programs will help me in engineering
		10. Knowing how to code computer programs will help me in science
		11. I believe I can be successful in coding

this term. The purpose of the interview during Study 1 was to glean students' interpretation of the items. In other words, we needed to know to what extent we and the students had the same understanding of these terms.

4.4.1 *Item 1: I Like to Imagine Making New Computer Programs.*

Of the 33 students interviewed, eight responded to the probe "What are computer programs?" in Item 1 by noting games or apps, either directly (e.g., Facebook) or generally, "a program is something you play or do on a computer." Seven students shared coding examples or experiences. These responses included, "creations that people can create through coding" and "a series of code, strands of code that when put into a computer, the computer does it." The team shifted phrasing as noted in Table 3. The logic behind the rewording was that students might be prompted to connect coding with the creation of computer programs by seeing the phrase *coding new computer programs*.

4.4.2 Item 3: I Am Good at Building and Fixing Computer Programs. Item 3 probed students on their self-concept of ability in building and fixing code. Students in this study were only asked to supply their answer—strongly disagree through strongly agree—to this item. Of the 33 students queried, 4 students freely offered that they had different answers for each action. These students considered them distinct processes or skills, thus, we opted to split this item into two as a result.

4.4.3 Item 8: Knowing How to Use Math and Science Will Help me to Create Useful Computer Programs. Forty-two percent of students simply restated the item or responded in general terms (i.e., "the basics of math and science will help me learn more") when asked if math and science would help them create useful computer programs. Seven students provided examples of the ways they experienced, or could imagine, math helping in coding. Student responses included, "Math are the variables, science.... I don't know." Only two expressed their connections to science as "technology" and "physics." To this end, the team determined to split the single item into different items with math, engineering, and science. In this way, additional probes could clarify students' understanding of the relationship of these subjects to CS.

4.4.4 Item 9: I Believe I Can be Successful in Computer Science and Programming. When prompted to answer, "What is computer science?" students were fairly evenly distributed; 14 answered "I don't know," whereas 10 answered by making programming or coding connections. The remaining students made general statements about how computers work (i.e., "what makes a computer work") or statements focused on the word science (i.e., "science on computers, different science, not normal science"). The team shifted wording on all items to only include coding. Our thinking was that this term reflected processes the students most likely encountered, through Hour of Code activities, for example.

4.5 Discussion

In summary, students struggled to understand *computer programs* and *computer science*, often confusing these concepts with general computer usage and with specific applications like Facebook and with computer games. As such, we determined that *coding* captured the essence of our interests and that young students were more likely to be familiar with, and have an understanding of, this word. Furthermore, our initial results indicated that approximately 75% of students did not understand how math and science together were connected to coding. We therefore made the decision to split the single item into three different items (math, science, and engineering). This was done to reduce extraneous cognitive load [40], as students would have to consider math and science individually, then together, and finally to consider how they may foster their use of coding. The team added the third term—engineering—as a way to assess to what extent young students understood what engineering is and how processes involved in this practice might align with computer science. Last, despite the fact that only a few students expressed that building and fixing code were distinct skills, we split the item into two. Our goal in doing so was twofold: one, to reduce cognitive load as noted above; and two, to elicit more detailed information from students by probing on the nuances between building code and debugging it. The changes to the items that we made following Study 1 appear in Table 3. We must note that we dropped an item—"I am curious about how computer programs work"—from Study 1, having found that students repeatedly asked the interviewer "didn't I already answer that?" The wording was very similar between two items (Items 4 and 6), so we opted to retain the one that appeared to be less confounding for students.

Table 4. Study 2 School-level Demographics

School	Black	White	Latinx	Other	NSLP*
Franklin	18%	55%	22%	5%	36%
Atwell	36%	29%	33%	2%	98%

*NSLP denotes students eligible for Free and Reduced Lunch.

5 STUDY 2

5.1 Participants

Our participants were 31 upper elementary students (ages 9–11) in two different schools in the southeastern United States. Table 4 presents demographic data on the schools and students that participated. The school names provided are pseudonyms to preserve anonymity. We returned to Atwell School to conduct cognitive interviews. Again, Atwell is a rural school with roughly equivalent percentages of African-American, Caucasian, and Hispanic/Latinx students. Franklin School is a suburban school with over 50% Caucasian and approximately 20% each of African-American and Hispanic/Latinx students. Students in Study 2 did not participate in any CS-specific intervention, and the interviewed students at Atwell did not participate in our previous intervention nor in earlier interviews. It was important for the development of appropriately worded items for us to query students from diverse socio-demographic backgrounds. There were 22 students interviewed at Franklin, 7 of whom were girls; 11 students were interviewed at Atwell, 6 of whom were girls.

5.2 Methods

In Study 2, the students read the items aloud, and the entire interview was audio recorded and transcribed. The interviews took approximately 8 to 15 minutes. Over the course of the interviews, the students were asked if any words were confusing or if they did not know the meaning of a word or phrase. As in Study 1, due to time constraints in the classroom, a subset of items was chosen for cognitive interviews. These items were chosen based on changes and findings from Study 1. These will be discussed in detail below. The purposes of the interviews shifted from Study 1; here, we were most interested in asking the students why they selected the answers they did. More specifically, we wanted to see in what ways students' chosen Likert-responses aligned with or deviated from their open responses.

5.3 Analysis

Similar to Study 1, after all the interviews were completed and transcribed verbatim, we thematically analyzed students' responses [10] for five items. The first author generated initial codes by reading through students' responses and pulling out salient phrases or words. Other authors then peer-checked [26] the initial codes and combined them to form themes. These themes were continuously checked against the data and to scan for patterns. Consensus was reached for all coding decisions. Additionally, two members of the research team assessed the students' responses for cognitive validity, rating the students' overall understanding of the item on a 0 to 4 Likert scale. Given that the ratings should be considered ordinal and not continuous, we calculated polychoric correlations [29] to assess rater agreement. The initial level of agreement ranged from 0.53 to 0.82. For items for which there was disagreement, consensus was used to reach agreement.

5.4 Findings

5.4.1 Item 1: I Like to Imagine Coding New Computer Programs. The student responses to this item broadly fell into four themes according to the word or phrase on which students focused. Those who privileged *liking* responded by noting, "I just like to play around" and "I don't like technology like that." Other students highlighted the phrase *to imagine* in their responses by stating, "I don't imagine about computers… [imagine means] fake things and … different things that might happen in the future." A third group of students focused on *coding* and noted, "Because I feel like when you're coding, you're in a whole different world and like you are in charge of whatever you're coding and you do what you want to do," and "Because, … I don't think I'm going to be a computer programmer, a coding person." A final group—*computer programs*—spoke of "not really [being] big on computer programming and stuff. I'm more an outdoors person" and "I like to imagine coding new computer programs because new computer programs can help people… like [they] can make things easier to access."

5.4.2 Items 3 and 4: I Am Good at Building/fixing Code. Students were asked to respond to these probes as individual items; however, their replies together highlight intriguing understandings of how elementary students consider the process of working with code. One student, for example, noted that she somewhat disagreed with being good at building code because "if I had to code by myself I'd probably get halfway through and just stop because I'm too impatient"; however, she stated she somewhat agreed with being good at fixing code because, "when someone messes up in code, I can fix it." Another student selected strongly agree for building code noting "we had to make a game… and I had to find out how to make all of it." He chose neither agree nor disagree for fixing code and explained his selection by stating, "some of the time I can fix the code… [and]other times, I keep looking and… I can't fix it." Students from Atwell School selected the disagree options more often (approximately 64%) than the Franklin School students (approximately 18%). Many of the Atwell students shared that they did not know what building nor fixing code meant and one noted that their teacher "only taught doing it, not like fixing it." Moreover, of the 31 students queried on these split items, 15 offered different answers for each.

5.4.3 Item 9: Knowing How to Code Computer Programs will Help me in Engineering. The most noticeable deviation in student responses to this item occurred between schools. Franklin School—with its weekly digital technology class for every homeroom—had student explanations such as "I feel like engineering has a lot to do with technology. And these days coding has a lot to do with technology as well" and "Because engineering is almost the same thing as coding. Except engineering is making the actual thing and coding is telling… that thing [what] to do." Atwell School—where there was more limited access to computers and technology-related activities—had student responses such as "I don't do engineering. We haven't learned that yet" and "In engineering, you work on cars." All students in Study 2 were asked to provide a definition for engineering. Franklin School students' definitions included: inventing and building, robots and technology, and building cars. Over 50% of Atwell students offered a definition that included "fixing cars."

5.4.4 Item 10: Knowing How to Code Computer Programs will Help me in Science. Students' responses to this item fell into three major themes: *science is technology, science is hands-on*, and *general computer use.* More specifically, *science is technology* students explained the following: "Science involves a lot of technology… It's like math, it's like a lot of different things combined like technology and math" and "Scientists have to code robots and that coding… means it's telling it what to do. Also, there's different types of sciences [and] one of those types do stuff with electronics." *Science is hands-on* responses included, "I mean science is like real life stuff and how to make chemical reactions and stuff like that. And coding is how to work with computers and make them work" and "Hm, 'cause I think of science as like putting different stuff together and testing things and learning about rocks and minerals and stuff. I wouldn't really use coding for science. I'd rather mix stuff together, make new things, and go outside and study." *General computer use* responses included, "Coding can help people when it comes to learning how to do stuff, when it comes to being online."

5.5 Discussion

For item 1 ("I like to imagine coding new computer programs") students did not focus on the terminology and intent we expected. We concluded that the wording of the item was unclear; students interpreted it in one of several ways rather than as a singular probe of whether they envisioned themselves using code to develop something new. We did not anticipate the word "imagine" would be problematic for the students as we conceptualized it to be an expression of interest. This highlighted the uncentered wording of the item and perhaps a vocabulary disconnect. Beers [8] suggests that active and effective readers use context clues and their prior knowledge to monitor their understanding of a text. Students concentrated on the concepts on which they could pull from experience or offer an appropriate response. The focus of some students on "coding" and others on "computer programs" is also important to note. Further investigation is required to understand whether the students in either category are able to connect both of those terms and whether they understand each of these terms thoroughly. Because of the overall finding of uncentered wording, the research team opted to reword the item entirely.

Modifying the original item 3 from a singular probe to two distinct items in Study 2 was appropriate, as approximately half of the students queried made distinctions between their self-concept of ability in building versus fixing code. Our findings are consistent with previous work, which concluded that debugging is a different set of skills than programming [1, 11, 41]. Our finding highlights the fact that students' actual and perceived abilities in this area should be considered separately. As such, the research team felt comfortable with the wording of these items and keeping them as distinct probes.

Item 9 appears to highlight disparities in schema development [27]. Overall, one group of students had a more robust understanding of engineering as a practice and could better conceptualize of how coding and engineering are synergistic skill sets and future occupations. In alignment with sociocultural theory, students arrive at school with widely varying home and community experiences and then receive additional distinct opportunities once there. Thus, schools and communities provide students access to both common and unique sources of culturally valued information, artifacts, and resources [4, 33, 44]. Disparities may occur due to unequal access to the cultural and technological resources because of inequitable funding structures or geographic limitations. Children bring both differing vocabularies and understandings of words to school. Our results highlight this; students from the suburban school (Franklin), in what is considered a "high-tech" employment region, had more developed understandings and experiences from which to draw and respond, whereas students from Atwell—situated in an under-resourced rural community—likely had fewer such experiences and therefore were unable to make the relevant connections to engineering as a diverse field of study.

Item 10 illustrates both the sociocultural context at play and a disconnect of educational policy and practice from public understanding. Science is an expansive domain, with unclear boundaries, and often is misrepresented and misunderstood [15]. Efforts to bring science into the public sphere—to make it interdisciplinary, more accessible, or immediately relevant—include the use of engaging and timely socioscientific issues [SSI; 50] in classroom instruction. Student responses appear to support the use of such pedagogical approaches in conjunction with a computer science curriculum to help them move beyond stereotypical or superficial interdisciplinary connections between science and coding.

In general, the term *coding* appeared multiple times throughout students' responses when they are talking about CS. Students' past experiences in both formal and informal environments can

Table 5. Study 3 Demographics

Context	Black	White	Latinx	Other	NSLP*
Summer Camp‡	31%	30%	13%	25%	N/A
Harris School	20%	54%	16%	9%	28%

‡Demographics are for participants, not for the entire camp.
*NSLP denotes students eligible for Free and Reduced Lunch.

contribute to the development of a conceptual understanding [32]. The fact that students' understanding of computer science often revolved around how or if they could relate coding to CS would be an outcome of previous experience in coding. Fundamental terms in computer science like this can thus be utilized in formal instructions to help students with CS concept development and domain identification. The research team felt comfortable with the wording of items 9 and 10, as the students were able to respond to the probes appropriately. However, we wished to gather more qualitative data on how students perceived the connections between CS and these other subject areas. Our hope was to begin to outline pedagogical implications based on students' experiences. To this end, we continued to query students on these items.

6 STUDY 3

6.1 Participants

Our participants were 32 elementary students (ages 8–11) in two different contexts in the southeastern United States. Table 5 presents demographic data on the contexts and students that participated. The first context was a summer camp associated with the university and intended for rising third- through fifth-grade students. The second context was a suburban school called Harris. Harris' student population is approximately 50% Caucasian, 20% African-American, and 15% Hispanic/Latinx. Student participants at Harris were in fifth grade and were pulled from their computer science class to participate in the interviews. The combination of the contexts and return rate of consent forms resulted in this study being disproportionately male. There were 18 students interviewed at the summer camp, 5 of whom were girls; 14 students were interviewed at Harris, 6 of whom were girls.

6.2 Methods

In Study 3, the students read the items aloud and the entire interview was audio recorded and transcribed. The interviews took between 5 and 15 minutes. Over the course of the interviews, the students were asked if any words were confusing or if they did not know the meaning of a word or phrase; one interview was ended early by the interviewer, as the child was unable to describe "coding." As in the earlier studies, due to time constraints in the classroom and camp activities, a subset of items was chosen for cognitive interviews. These items were chosen based on changes and findings from Study 2. These will be discussed in detail below. The purposes of the interviews shifted from Study 2; in the current study, we were most interested in eliciting from students how they understood the wording of the item as well as why they selected the answers they did.

6.3 Analysis

Mirroring analyses from Study 1 and 2, once the interviews were transcribed verbatim, the first two authors generated initial codes and collapsed them into themes, which were peer-checked by others on the research team. Consensus was reached for all coding decisions. This thematic analysis occurred for three items. Similar to Study 2, we calculated polychoric correlations [29] to assess rater agreement. The initial level of agreement ranged from 0.54 to 0.80. For items for which there was disagreement, consensus was used to reach agreement.

6.4 Findings

6.4.1 Item 1: I Like to Use Coding to Make Something New. The majority of the responses fell into the theme of *Item Reiteration* (N=18). Responses that fell under this category include, "If you would use coding and make something new out of it…" Under *Item Reiteration*, we separated responses under the subcategories *Close Reiteration* (N=9) and *New* (N=15). Those in the second subcategory contained sentences that focused on the word "new." An example of an answer that contained both is, "It is asking me to use different types of coding to make something completely new out of that coding. Like making a new program that can answer something that no one else has really answered." The first sentence in the response is a *Close Reiteration* and the second is a deeper explanation that focuses on "new."

Two other themes that emerged from the students' answers related to this item were about the *Goal of Coding* (N=9) and the students' *Attitudes Towards Coding* (N=2). An example of a response containing goals of coding is, "To make some kind of new program. So programming a game or a… or something where kids can use that coding to learn how to code in Scratch." The attitudes towards coding include, "That you like coding" and "It means that if you really want to make something, like a new invention or something, you can use coding to do it and that would be pretty cool."

The remaining responses were unlike those noted above. Some of them contained examples of specific blocks (e.g., *move* blocks) that the students likely used in classes. Other interesting responses were, "That um, I wanna like, try new things and [do them] through coding" and "It means using coding, you can create something and that is… new in the sense that it's coming from you…" We found these responses particularly interesting, because the students were focused on trying new things, and the new things were their creation. The second quotation implies that the student feels ownership over his/her creation.

6.4.2 Item 9: Knowing How to Code Computer Programs will Help me in Engineering. Students' responses to this item fell into two large themes—*Career* and *Item Reiteration.* Career-based responses (N=11) included mentions such as, "It's asking if coding, if you want to be an engineer, it will help you with your job" and "Like [coding] will help you when you're engineering stuff if you become an engineer." Item reiteration responses (N=14) were simply instances in which students restated the item but did so by substituting their own words and/or by providing explanatory examples. Such examples include "I think it's asking me, like, how… coding computer programs help you in like, like, engineering … like how to build stuff…" and "It is asking me if knowing how to code the computer, create programs, or games, or websites, would help me in engineering when I'm building something."

Student responses also resulted in other themes, fewer in number, but worthy of note. These included *Math*; *Coding in Engineering*; and *Cars, Robots, and Technology.* The math-based responses included this statement: "Well code can help you learn math because you have to be able to use math to code sometimes. And in engineering you have to use math." *Coding in Engineering* responses included pronouncements such as, "How coding, if you know how to code you can know how to engineer stuff" and "It is asking if computer programming will help in engineering for people, and how it will maybe benefit them." The last theme—*Cars, Robots, and Technology*—is an aggregate of examples students offered for how they see coding directly applying to their understanding of engineering. For example, "So if you're engineering, … you would need um code- coding 'cause sometimes you can make like any type of technology maybe. Anybody can make like a hovering car, like a real-life hoverboard. Like you would need coding for that."

6.4.3 Item 10: Knowing How to Code Computer Programs will Help me in Science. Students' responses to item 10 fell into three themes, although the vast majority of students (N=22) reiterated the item by rewording it and/or providing an example. Such *Item Reiteration* responses include, "It's asking me that in science, knowing how to code will be helpful" and "How code can help you in science… and how you can use computer programs to help you understand what science is better." Of note, only two students mentioned *Careers* in their response to this item: "It's asking you if coding computer programs will do something if I like to do science… Because I am going to have to use a lot of science when I grow up because I'm going to have multiple jobs, I think." The final theme is an aggregate of how students *Connect Science to Coding*. Three students explicitly noted that they do not see how coding and science connect. One such response was "Because [in] science you learn about volcanoes and how they work and I don't think coding really involves science." The remaining responses in this theme show alignment between science and coding, albeit in varied ways. For example, "Like, if you know how to code a computer, how it will help you in science… Because, um, it has a little bit of science to it, and it also has like experimenting in it to see what happens, and that's kind of part of science."

6.5 Discussion

For Item 1, over half the students queried offered item-reiteration responses. The majority of these responses focused on the word "new" in the item. Because of the number of students that seemed to understand the intent of the item, we believe that the wording of the item is appropriate for upper elementary students. It is also important to note that under *Goal of Coding*, the majority of the responses (N=7) contained sentences about games. The students that spoke about games seemed to have trouble coming up with any other specific examples of new programs they can create, "Like, to use coding, coding is a computer thing to use on a computer to make a game or any, to make characters move or to create something." This reflects the types of activities that students associate with coding. To better help students understand that they can complete a wide variety of tasks by coding, practitioners and researchers should focus on curricula and activities that are more reflective of problems computer scientists solve.

Regarding item 9, just under half the students queried offered item-reiteration responses. However, many of the career-themed responses also restated the item in such a way that students clearly understood the intent of the item, as we hoped. As such, we feel confident moving forward with this item as worded. It is important to note that this item, more so than item 10 below, had more career-themed responses. We posit that elementary students do not typically take classes in engineering as they do in science or math and may not conceptualize it as being anything other than an activity that occurs distally, as an adult.

Students' responses to item 10 largely support our intent in writing the item. The students were able to connect coding and science in ways that highlight how the skills learned through coding could help their learning in science. It is important to note that students' understanding of science is varied; they offered definitions that ranged from "the learning of everything" and "think[ing] of new stuff and new ways to help people" to the more specific topics within science such as chemistry, planetary studies, force and motion, and the human body.

Across all three items, several responses are worth highlighting. One student suggested that coding seemed more appropriate in an English language arts class: "[It's] like learning new words or fixing things I guess. Let's say I was writing a summary and I wanted to fix some things, and that is technically coding because I am fixing things that I messed up."

Another student conflated cryptography and coding, noting at one point and in response to item 1, "I guess to like, uncode an answer, and to use different codes to make like words. Or you can do different things with codes." In response to item 9, this student offered this: "It is asking me to figure out a way to like engineer different, like, machines that can take in codes and decode them or you can code one then decode it and code it again." We fully recognize that cryptography is an important topic under cybersecurity within computer science; that a student has conflated these concepts underscores the need for the CS education community to more fully utilize appropriate terminology within CS activities for young students.

Moreover, two students expressed the benefits of using coding as a planning and modeling tool to help with the *doing* of science or engineering. One noted, "Engineering is one of those things where you need the 3D model- modeling and you've got a lot of math involved in that." The other stated, "Well engineering is kind of like designing things and making things better, even just making a new invention. And you can use coding to make it work on there before you actually start it because if you actually start it and you don't use coding then it will be hard to plan it out, I guess. And if you don't use coding or a plan to start it before you actually do it then if you make mistakes you can't fix it on the coding."

7 FINAL DISCUSSION

Cognitive interviews are a potent tool for systematically investigating children's self-reports with an acceptable cognitive validity [49]. In the process of developing this instrument, we analyzed multiple aspects of the students' responses. These include the students' self-concepts regarding CS problem solving, their understanding of CS as a domain, perceptions of other related domains, their understanding of CS-specific concepts like fixing/debugging, and the associated prompts as well (i.e., "what is engineering?"). Overall, our findings reinforce the importance of revalidating instruments when adapted to new foci or used with younger audiences. If we had simply taken the S-STEM instrument, designed for middle-grade students with an engineering & technology focus, and adopted it for elementary students with a CS focus without this cognitive interviewing process, we would have likely had psychometrically problematic results. As an added benefit, our studies provided important insights into children's thinking around core CS concepts, thus informing both curriculum development and pedagogical strategies.

Our findings support our view that prior experience and opportunities afforded to students shaped their responses as much as their general developmental level. Piagetian theory supports that children in the 9-to-11 age range can think about and solve problems that pertain to real, or actual, objects [31]. This may well be why we saw some marked differences in responses between the schools. Some students had actual experience with concepts about which they were probed, whereas others had no such experiences. This lack of experience would have been too abstract for this developmental phase. One immediate implication of this is the need to imbue elementary curricula with terminology and experiences that connect with and transcend what the children encounter in their own communities. Our findings are in alignment with both sociocultural theory in general [33, 44] and social capital theory as applied to STEM areas [4]. At the policy level, our findings reinforce stated concerns about the opportunity gap for youth with regards to exposure and cultivating interest in high-value STEM career pathways [14].

It is important to reconsider and reword survey items when exploring specific domains. Our expert conceptions of appropriate language do not always correspond with young students' understandings of terminology. For example, over the course of these studies, item 1 underwent several important changes. The original S-STEM [43] Technology and Engineering wording was "I like to imagine making new products." The team considered "I like to imagine making new computer products" to adhere as closely as possible to the original. This was rejected in favor of "I like to imagine making new computer programs." Still, students had difficulty with this wording, as they could not isolate what computer programs were. In hopes of probing their thinking, we shifted wording slightly to "I like to imagine coding new computer programs." Students then grappled unnecessarily with this item, as they did not focus on what we hoped: coding for creation. As such, we shifted wording to "I would like to use coding to make something new."

The language educators use with students is exceedingly powerful. Helping students broaden the connections between their everyday and scientific language may prove influential. One such example is in our use of "fixing" as a synonym for "debugging." One student offered the following explanation of fixing: "I can code and I think that fixing code is probably a little harder because it includes understanding the code and then being able to change it and make it better, or fix something that is wrong with it." Students did not often share such profound understandings of fixing/debugging. Care needs to be taken to ensure that students' comprehension of essential CS concepts straddles not only diverse socio-demographic school contexts but also from primary to intermediate to secondary education levels. Some students' domain understanding of subjects such as engineering and science—and the processes involved in these subjects—are still blurry to elementary students. Because these terms are so interconnected as well as important to 21st century learning, curriculum writers and educators need to focus on how to make the domain-specific terminology clear.

8 LIMITATIONS

Findings of these studies should be considered in the context of the following limitations: The first is sample size and methodology. Cognitive interviewing is time-intensive in nature; therefore, we opted to prioritize quality over quantity. As such, we purposefully selected contexts that reflected a diverse array of student backgrounds to capture a range of student responses and experiences. The second limitation is our decision to probe students on only a select number of items. This likely could have been remedied by conducting the studies in a lab setting; however, we chose natural learning environments to open the study to as many students as possible. The third limitation is in reference to the socio-demographic variables we were able to collect; future work might consider collecting more variables. Last, some of the student participants expressed that they had never been interviewed, and others shared that they had never been asked questions about their understanding of a statement. The cognitive interviewing process was new for all the students and may have caused some discomfort for some who felt they needed to share a *right* answer with the researchers.

9 CONCLUSIONS AND FUTURE WORK

Over the course of these three studies, we qualitatively analyzed students' responses to the wording of 13 items, resulting in a final set of 11 items deemed appropriate for upper elementary students. We intend to distribute our survey to quantitatively validate the instrument. As such, we anticipate this instrument being used to measure fourth- and fifth-grade students' attitudes toward CS, in particular before and after completing a relevant intervention.

There is still much to understand about how elementary students learn computer science. Because their perceptions of CS can affect their learning, it is important for researchers in the computer science education community to study what those perceptions are, how they came to be that way, and the importance of addressing any misconceptions in a manner that can broaden participation in the field. Our results contribute to the CS literature by showing the varied ranges of conceptions young students have regarding these concepts. As can be seen from their responses to probes of their thinking, upper elementary students often have somewhat vague understandings of core and essential vocabulary such as computer science and programming. Additionally, young students have diverse notions of how CS may connect with other subjects they learn. Our results are also of pragmatic curricular interest, as they highlight voids in instruction regarding not just key vocabulary, but perhaps of specific CS processes like debugging.

Students' varied conceptions of how CS/CT and other STEM subjects might align is in agreement with prior work on student conceptions of how STEM academic areas relate to each other and to future career pathways [48]. These findings point to any number of potential interventions to be implemented and studied. Of particular interest would be if students' awareness changes after participating in a researcher-led intervention specifically designed around CS/CT integration into traditional STEM academics (e.g., science and mathematics).

The computer science education community may consider taking up research using this systematic approach to cognitive interviewing to ensure measures of young students' attitudes and perspectives are cognitively valid. There are several potential directions to consider. For example, we may be interested in knowing to what extent the students self-regulate their learning—how and if they plan, monitor, and evaluate—in a coding environment. Moreover, including an open-ended prompt, allowing students to offer any final thoughts, may elicit rich information about which we never thought to inquire. Finally, querying students on their attitudes toward collaborative coding may indicate potential roadblocks and solutions to encourage students to work together on coding activities. Cognitive interviewing is a powerful tool for researchers interested in developing more stable and reliable instruments. Moreover, curriculum developers, who wish to create materials that provide students with authentic learning experiences that help to bridge their existing understanding with new content, may find it helpful as well. As seen from these three studies, it was an important tool to capture students' understanding of CS language and processes.

ACKNOWLEDGMENTS

We thank the teachers and camp counselors who provided us time and space to conduct our interviews, and the students who devoted their time and energy to provide thoughtful responses to our questions.

REFERENCES

[1] M. Ahmadzadeh et al. 2005. Novice programmers: An analysis of patterns of debugging among novice computer science students. *Inroads* 37, 3 (2005), 84–88.

[2] I. Ajzen and N. Gilbert Cote. 2008. Attitudes and the prediction of behavior. In *Attitudes and Attitude Change*, W. D. Crano and R. Prislin (Eds.). Psychology Press, New York, NY, 289–311.

[3] J. Alano et al. 2018. K–12 Computer Science Framework. Retrieved from https://k12cs.org/.

[4] L. Archer et al. 2015. "Science capital": A conceptual, methodological, and empirical argument for extending Bourdieusian notions of capital beyond the arts. *J. Res. Sci. Teach.* 52, 7 (2015), 922–948. DOI: 10.1002/tea.21227

[5] M. W. Arthur et al. 2002. Measuring risk and protective factors for use, delinquency, and other adolescent problem behaviors: The communities that care youth survey. *Eval. Rev.* 26, 6 (2002), 575–601. DOI: 10.1177/019384102237850

[6] R. M. Baars et al. 2005. The European DISABKIDS project: Development of seven condition- specific modules to measure health related quality of life in children and adolescents. *Health Qual. Life Outc.* 3, 1 (2005), 70. DOI: 10.1186/1477-7525-3-70

[7] R. S. Baker et al. 2010. Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive–affective states during interactions with three different computer-based learning environments. *Int. J. Hum.-Comput. Stud.* 68, 4 (2010), 223–241.

[8] K. Beers. 2003. *When kids can't read, what teachers can do.* Heinemann, Portsmouth, NH.

[9] P. Bourdieu. 2011. The forms of capital (1986). In *Cultural Theory: An Anthology*, I. Szeman and T. Kaposy (Eds.), John Wiley, Chichester, UK, 241–258.

[10] V. Braun and V. Clarke. 2006. Using thematic analysis in psychology. *Qualit. Res. Psychol.* 3, 2 (2006), 77–101. DOI: 10.1191/1478088706qp063oa

[11] K. Brennan and M. Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the American Education Researcher Association Conference.*

[12] D. A. Brodersen and G. C. Thornton. 2011. An investigation of alpha, beta, and gamma change in developmental assessment center participants. *Perf. Improv. Quart.* 24, 2 (2011), 25–48. DOI: 10.1002/piq.20109

[13] L. Carter. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bull.* 38, 1 (2006), 27–31.

[14] Code.org and CSTA. 2018. 2018 State of Computer Science Education: Code.org. Retrieved from https://advocacy.code.org/.

[15] N. W. Feinstein. 2015. Education, communication, and science in the public sphere. *J. Res. Sci. Teach.* 52, 2 (2015), 145–163. DOI: 10.1002/tea.21192

[16] E. A. Forman and J. Larreamendy-Joerns. 1998. Making explicit the implicit: Classroom explanations and conversational implicatures. *Mind, Cult. Activ.* 5, 2 (1998), 105–113. DOI: 10.1207/s15327884mca0502_4

[17] P. Gibbons. 2013. *Scaffolding Language, Scaffolding Learning: Teaching ESL Children in the Mainstream Classroom.* Heinemann, Portsmouth, NH.

[18] A. C. Graesser et al. 2006. Detection of emotions during learning with AutoTutor. In *Proceedings of the 28th Annual Meeting of the Cognitive Science Society.* 285–290.

[19] S. Grover et al. 2014. Remedying misperceptions of computer science among middle school students. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education.* ACM, 343–348. DOI: 10.1145/2538862.2538934

[20] M. Hewner. 2013. Undergraduate conceptions of the field of computer science. In *Proceedings of the 9th Annual International ACM Conference on International Computing Education Research.* ACM, 107–114. DOI: 10.1145/2493394.2493414

[21] M. Hewner and M. Guzdial. 2008. Attitudes about computing in postsecondary graduates. In *Proceedings of the 4th International Workshop on Computing Education Research.* ACM, 71–78.

[22] V. John-Steiner and H. Mahn. 1996. Sociocultural approaches to learning and development: A Vygotskian framework. *Educ. Psychol.* 31, 3–4 (1996), 191–206.

[23] S. A. Karabenick et al. 2007. Cognitive processing of self-report items in educational research: Do they think what we mean? *Educ. Psychol.* 42, 3 (2007), 139–151.

[24] V. Kukul et al. 2017. Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Educ. Technol.-Theor. Pract.* 7, 1 (2017), 158–179.

[25] C. M. Lewis et al. 2016. I don't code all day: Fitting in computer science when the stereotypes don't fit. In *Proceedings of the ACM Conference on International Computing Education Research.* ACM, 23–32. DOI: 10.1145/2960310.2960332

[26] Y. S. Lincoln and E. G. Guba. 1985. *Naturalistic Inquiry.* Sage, Newbury Park, CA.

[27] M. B. McVee et al. (2005). Schema theory revisited. *Rev. Educ. Res.* 75, 4 (2005), 531–566.

[28] National Science Foundation. 2019. STEM + Computing K–12 Education (STEM+C). Retrieved from https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505006.

[29] J. C. Nunnally. 1978. *Psychometric Theory.* McGraw-Hill, New York, NY.

[30] Ö. Özyurt and H. Özyurt. 2015. A study for determining computer programming students' attitudes towards programming and their programming self-efficacy. *J. Theor. Pract.* 11,1 (2015), 51–67.

[31] J. Piaget. 2002. *Judgement and Reasoning in the Child.* Routledge, London.

[32] A. L. Pines and L. H. West. 1986. Conceptual understanding and science learning: An interpretation of research within a sources-of-knowledge framework. *Sci. Educ.* 70, 5 (1986), 583–604.

[33] P. R. Portes and J. A. Vadeboncoeur. 2003. Vygotsky's educational theory in cultural context. In *Mediation in Cognitive Socialization: The Influence of Socioeconomic Status*, A. Kozulin, B. Gindis, V. Ageyev, & S. Millier (Eds.). Cambridge University Press, Cambridge, UK, 371–392.

[34] S. Psycharis and M. Kallia. 2017. The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instruct. Sci.* 45, 5 (2017), 583–602. DOI: 10.1007/s11251-017-9421-5

[35] R. C. D. Reis et al. 2018. Affective states in computer-supported collaborative learning: Studying the past to drive the future. *Comput. Educ.* 120 (2018), 29–50.

[36] N. C. Schaeffer and S. Presser. 2003. The science of asking questions. *Ann. Rev. Sociol.* 29, 1 (2003), 65–88. DOI: 10.1146/annurev.soc.29.110702.110112

[37] N. Schwarz. 1999. Self-reports: How the questions shape the answers. *Amer. Psychol.* 54, 2 (1999), 93.

[38] N. Schwarz. 2007. Attitude construction: Evaluation in context. *Soc. Cog.* 25, 5 (2007), 638–656.

[39] P. Sengupta et al. 2013. Integrating computational thinking with K–12 science education using agent-based computation: A theoretical framework. *Educ. Inf. Technol.* 18, 2 (2013), 351–380. DOI: 10.1007/s10639-012-9240-x

[40] J. Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cog. Sci.* 12, 2 (1988), 257–285.

[41] Y. Tran. 2019. Computational thinking equity in elementary classrooms: What third-grade students know and can do. *J. Educ. Comput. Res.* 57, 1 (2019), 3–31. DOI: 10.1177/0735633117743918

[42] M. J. Tsai et al. 2019. Developing the computer programming self-efficacy scale for computer literacy education. *J. Educ. Comput. Res.* 56, 8 (2019), 1345–1360. DOI: 10.1177/0735633117746747

[43] A. Unfried et al. 2015. The development and validation of a measure of student attitudes toward science, technology, engineering, and math (S-STEM). *J. Psychoeduc. Assess.* 33, 7 (2015), 622–639. DOI: 10.1177/0734282915571160
[44] L. S. Vygotsky. 1980. *Mind in Society: The Development of Higher Psychological Processes.* Harvard University Press, Cambridge, MA.
[45] D. Weintrop. 2016. Modality matters: Understanding the effects of programming language representation in high school computer science classrooms. *Doctoral dissertation.* Retrieved from http://www.terpconnect.umd.edu/~weintrop/papers/WeintropDissertation.pdf.
[46] D. Weintrop et al. 2016. Defining computational thinking for mathematics and science classrooms. *J. Sci. Educ. Technol.* 25, 1 (2016), 127–147. DOI: 10.1007/s10956-015-9581-5
[47] A. Wigfield and J. S. Eccles. 2000. Expectancy—Value theory of achievement motivation. *Contemp. Educ. Psychol.* 25, 1 (2000), 68–81. DOI: 10.1006/ceps.1999.1015
[48] E. Wiebe et al. 2018. The relationship of STEM attitudes and career interest. *EURASIA J. Math. Sci. Technol. Educ.* 14, 10. DOI: 10.29333/ejmste/92286
[49] M. E. Woolley et al. 2004. Cognitive pretesting and the developmental validity of child self- report instruments: Theory and applications. *Res. Soc. Work Prac.* 14, 3 (2004), 191–200. DOI: 10.1177/1049731503257882
[50] D. L. Zeidler. 2014. Socioscientific issues as a curriculum emphasis: Theory, research and practice, S. K. Abell and N. G. Lederman (Eds.). In *Handbook of Research on Science Education.* Routledge, Taylor and Francis, NY, 697–726.

Received August 2019; revised December 2019; accepted February 2020