The Collaborative Learning Framework: Scaffolding for Untrained Peer-to-Peer Collaboration

Brittany Ann Kos
University of Colorado Boulder
Boulder, CO
brittany.kos@colorado.edu

ABSTRACT

Recently, we've seen huge enrollment increases in computing and technology courses, which make it difficult for instructors to work personally with students and give them individualized instruction. Instructors may encourage students to work together, but we cannot assume that students know how to ask for assistance from their peers or that students will know how to provide meaningful educational support. Our motivation for this study was to create a computer science educational intervention that provided students with a scaffolded framework that helps them work through problems together. Our goal was to create a simple and quickly understood resource for students to use during collaborative lab activities. This paper describes the "Collaborative Learning Framework" (CLF), an educational support tool operationalized as a poster. The CLF poster is intended to be hung in a classroom or lab space and includes four prompts that ask students to explain and think critically about their problems with their peers. In Summer 2017, we conducted an exploratory qualitative study of an introductory, non-major computing course. Our findings present case studies of three CLF poster users and two non-CLF poster users. We evaluate the CLF poster by identifying how the students develop a computational thinking mindset and use the poster as an instructional problem-solving tool. We found that the CLF poster was an effective and useful collaboration tool for the students who were developing a computational thinking mindset.

KEYWORDS

collaboration, CS0, CS1, non-majors, information science, education

1 INTRODUCTION

Over the last few years, we've seen huge enrollment increases in computer science and computing courses. Large classes make it difficult for instructors to work one-on-one with students and give them individualized instruction. There have been trends to add lab or recitation hours to courses [11, 13], employing undergraduate teaching assistants [4–6], and creating help or tutoring rooms [8]; however, these measures can be costly and do not scale easily. Even when students have the opportunity to get individualized help from an instructor or graduate TA, students may not take advantage of this opportunity, because they may be intimidated by asking questions and seeking help from authority figures [5]. Some students may look to their classmates or students who have taken the class previously when they need help with an assignment. Students with similar backgrounds and experience levels often feel more comfortable seeking help from people they consider their equals rather than individuals they feel are more experienced and might judge their lack of skill [5]. Though this peer-peer interaction can benefit students, this method relies on untrained, novice programmers guiding each other. We cannot assume that students will know how to seek help from a peer or that they know how to give support that is educational and avoids telling each other the answers. Inexperienced students may also face challenges when differentiating between helping their peers solve problems themselves and answering problems for them. In computer science, it can be difficult to create clear boundaries between cheating and collaboration [12]. These ideas can be ill-defined in syllabi [1] and it is unfair to encourage students to work together without providing them clear boundaries between collaborating and cheating. Our motivation was to create a computer science educational intervention that provided students with a scaffolded framework that provided them with methods for effective collaboration. Our focus was to create a simple and immediately understood resource for students to use when they were working together during lab activities. We see this intervention as a supplement to other, ongoing computing educational efforts.

Working Paper, August 2017, Boulder, CO, USA

2 LITERATURE REVIEW

Our educational intervention builds off of the "Collaborative Discussion Framework", which encourages collaborative discussion between students and provides them with language and prompts that assist them in asking and giving help to their peers [10]. Previously, this framework has been used in elementary school classrooms to help students learn computing concepts and program in Scratch [9]. Though this framework was intended for younger audiences, it was designed for novice programmers to further develop their computational thinking. To adapt the collaborative discussion framework to an undergraduate audience, we pulled from the literature about Undergraduate Teaching Assistants (UgTAs). Universities have seen success training undergraduate students to work as teaching assistants in undergraduate classrooms for a number of years now [6]. UgTAs often have differing responsibilities depending on the university or course employing them [4], but they are primarily hired to offer additional tutoring or instructional help in a course [3]. UgTAs are often closer in age to students enrolled in a course or have taken the course more recently and are seen as less of an authority figure [5]. This makes them ideal candidates to help other undergraduates in a course because they can be a less intimidating and more relatable resource. For our study, we've specifically looked at the best practices and tips that instructors have suggested for training UgTAs and incorporated them into our study [7].

In parallel, we draw on the idea of computational thinking (CT) [14] to contextualize the analysis of our study. We use Brennan and Resnick's computational thinking framework, which acts as an operationalized definition of CT that describes what students learn while programming [2]. The computational thinking framework involves three dimensions: "computational concepts (the concepts designers engage with as they program, such as iteration, parallelism, etc.), computational practices (the practices designers develop as they engage with the concepts, such as debugging projects or remixing others' work), and computational perspectives (the perspectives designers form about the world around them and about themselves)" [2]. These three elements provide a method define CT as a student's actions, computational understanding, and metacognition and it provides a way to assess how students are developing their computational thinking throughout the course.

3 METHODS AND CONTEXT

We conducted a two-part exploratory study where we introduced the CLF poster in a non-major introductory programming class in the summer of 2017. The class was offered through the Information Science department and was a requirement for all of the students in the College of Information and Media, where the Information Science department is housed. We chose this introductory computing class because it was intended for inexperienced programmers and focused heavily on teaching computational thinking concepts while the students were introduced to Scratch and Python programming.

3.1 Creating the Intervention Tool

We use Park and Lash's Collaborative Discussion Framework [10] and draw on their pre-existing poster resources, which encourage students to collaborate on their programming problems. The CLF poster has a series of four questions that "provides students with language to assist them in seeking and giving help" [9]. We modified the language Park and Lash's questions, so they were easier to read in the classroom. These questions read: (1) "What are you trying to do?"; (2) "What have you tried already?"; (3) "What else can you try?"; and (4) "What would happen if...?" These are broad questions that help guide collaborative discussions about programming problems. The questions are designed to get students to assess their problems and explicitly describe what their goals and thought process is. We added additional subquestions that were paired with the four main questions that gave students additional ways they could discuss their situation. We chose not to number any of the main questions or subquestions since we did not want to create an artificial hierarchy or suggest students should work through the questions linearly. We also added language at the top of the poster that would quickly contextualize the content of the poster and tell students how to use it as a resource. We didn't want to lose students' interest by adding tedious text at the beginning of the poster, but we also needed to introduce the purpose of the poster and tell how it is intended to be used. Finally, we added some visual interest to the poster by adding images that were intended to be fun and relevant to the poster's content. The final CLF poster design is shown in Figure 1.

Figure 1: The Collaborative Learning Framework (CLF) Poster

3.2 Ethnographic Observation

We first introduced our intervention tool at the beginning of the second week of the five-week summer course. This was when the class was switching from Scratch to Python programming and starting to learn more difficult programming concepts. We had Wanda, the instructor of the course, introduce the poster as a tool as a regular part of her instruction. We did not want to introduce the poster ourselves in case we influenced the students. We had one researcher observe the class' interactions with the poster for the next three days. The first day of observation was a lecture day where Wanda taught booleans and conditionals. The second and third observation days were free lab days for where the students worked on their weekly project, a "Choose Your Own Adventure" game. (This is a narrative text-based game where the player is lead through the story through by selecting different options.) We chose to begin our study during this three day period because Wanda was introducing new material. This offered us the opportunity to gain insight into how students approach learning and programming new concepts and we could also assess how the CLF poster aided them in developing their computational thinking.

3.3 Focus Groups

After the three-day ethnographic observation period ended, we conducted two focus groups of two students each, held one week after the classroom observations. The focus group participants self-categorized themselves as CLF poster users and non-CLF poster users. Each student participant was compensated with a $5 coffee gift card for their participation. Caitlyn (F) and Sean (M) were non-CLF poster users, while Carrie (F) and Rebecca (F) were CLF poster users. All four students were fourth or fifth-year seniors, majoring in a media or communication field. None of the students were new to seeing code, Sean and Caitlyn had taken web design (HTML/CSS) classes and Carrie and Rebecca had taken statistics and economics courses that used the R programming language. In class and lab time, Carrie and Rebecca worked together with another female student, Lucy, while Caitlyn often worked alone. Sean was an undergraduate assistant in the course, his job was to help students during lab time, answer questions, and help students with their assignments; he had taken this course the previous semester. The next section discusses our findings from the three ethnographic observation days and two focus groups.

4 FINDINGS

The analysis of the classroom observations and focus groups brought out many interesting themes. There are many important similarities and differences in the ways that the students discuss their classroom experiences learning to code and working through their problems. This section outlines the three dimensions of Brennan and Resnick's computational thinking framework: Computational Concepts, Computational Perspectives, and Computational Practices [2]. We use this framework as an analysis tool that provides a theoretical structure in which to assess the student's development of computational thinking. In our analysis, we look for instances where the CLF poster enriches student experiences, guides them work through their problems, and helps them develop their computational thinking.

4.1 Computational Concepts

High-level and broad concepts–such as data storage, booleans, and conditionals–were easily understood by students on a theoretical basis. They were able to think about these concepts in their own terms; however, many students had trouble implementing these concepts and using them with algorithms. Sean talks about seeing this cognitive gap when he assisted students:

> *Sean: I think it's easier for a lot of people to grip the computation thinking than it is the syntax.*

From our observational data, we saw that nearly half of the questions we witnessed Sean helping students with stemmed from syntax errors. He also sees that students understand at a high-level what they want their program to do, but they have trouble actually writing the code. Rebecca also discussed this problem explicitly when she talked about working with dictionaries and arrays in Python and learning to use the AND operator to combine two booleans in a conditional statement:

> *Rebecca: I have a clear idea of what that is in my head but not in terms of coding.*
>
> *Rebecca: Initially I was trying to conceptualize it in a way that was already in my brain . . . this isn't necessarily going to match what I've known or how I've formerly done things.*

As Rebecca learns new programming concepts, she also recognizes that she needs to adapt her thinking to better fit with a computational perspective. She associates learning new computational concepts to needing to think in a computational way. This is an important step in developing her computational thinking. In their focus group, Caitlyn and Sean also talk about a time when Sean assisted Caitlyn with her "Choose your Adventure" game:

> *Caitlyn: I was just–I was stuck and he just kind of–I don't know how to describe it . . . I think I was just stuck in my own head like, "Okay, this code applies for this but then it's not the same for this." So I was like, "Shit is this a completely different format?" So that's when I'd call him over and he'd be like, "No, no, no, it's the same, you just have to reorganize the numbers and stuff."*
>
> *Sean: She had everything there, just not formatted correctly. So I obviously just tried to give hints and essentially help build it up because I would be like, "Oh this needs to be here because of this" and like "does that make sense?" Then logically work down.*

In the focus group, it was difficult to get both of them to describe Caitlyn's error in detail. In class observations, we observed that Caitlyn had ordered her nested her conditional statements in a way that caused the game logic to run in unintended ways. Caitlyn and Sean's ill-explained answers are signs of novice programmers. They both explain the problem with vague statements and use terminology that is familiar to them, not the computing phrases or terminology that has been taught in the class. If we focus on Sean's actions when he helps Caitlyn, he is seen almost explicitly telling Caitlyn how to debug her program. Though he is explaining his actions as he goes, he does not let Caitlyn work through her algorithm herself and he prevents her from being able to work out the problem herself. In this interaction, Sean is also leading the discussion and only asks for Caitlyn's answer to a yes or no question at the very end of his explanations. We see both students having trouble with novice level explanations of computing concepts and Sean dominating the conversation with Caitlyn and telling her and answers instead of helping her learn on her own.

4.2 Computational Perspectives

This section discusses the different metacognitive views of Caitlyn, Rebecca and Carrie as they develop their computational thinking skills. We asked Caitlyn to talk about her experiences have been like learning CT. Her response is unclear ad confuses learning computational thinking with learning other computational concepts during lecture:

> *Caitlyn: I don't know. Sometimes when she's lecturing I just–I take a break and then I come back. I don't know, the whole computational thinking thing it like helps because you forces your brain to go there but at the same time–I don't know . . . It's an interesting class. Thankfully this isn't a 3-hour class, it's an hour and*

a half. So you can walk away and sometimes I just go take a nap and then I'll wake up and I'll try it on my own, which a lot of times helps. Her classes are good but I usually just end up looking over the lecture slides later.

Though she was asked to reflect on her experiences learning about CT, Caitlyn instead talks about how unpleasant lectures are for her. It seems like she thinks that listening to lecture and learning computational thinking are closely related. It is unclear whether she conceptualizes CT and class lecture as separate and if her dislike of lecture may be influencing her thoughts about CT. When we asked her to describe how she thinks about problems, she describes the process different from a computational thinking process:

"I'm a straightforward thinker . . . I just want the answer and then I like to go on my own and figure out the why. I feel like I can figure it out on my own."

Caitlyn seems to have a very clear idea about how she likes learning. In both of her excerpts in this section, she mentions how she likes working alone and working through problems on her own terms. Caitlyn is not developing strong computational thinking skills, she is holding firm on to her own way of doing things. This will heavily influence her computational practices and will be a key factor in explaining her non-CLF poster use.

Now, we'll contrast Caitlyn's situation to Carrie and Rebecca's, who worked on developing their computational thinking and hold a clear computational perspective. Both women describe their shift towards a computational perspective:

Rebecca: I think for me since it's such a different way of thinking and conceptualizing different ideas or processes I think for me putting myself in a frame of mind that's different than what I'm used to, that has been a challenge for sure . . . I think initially I was trying to conceptualize it in a way that was already in my brain. Like, how can I make how I operate, and how my internal monologue operates, match this? Instead of really just trying to understand programming for what it is and how it works. So I think reconciling that relationship between-this isn't necessarily going to match what I've known or how I've formerly done things and jumping into something where I'm like, "okay this is a new way of thinking" and accepting it and kind of moving on.

Carrie: I think I'd like to add to that. I feel all that too, and to add to that, it's been really odd because I didn't realize that I wasn't thinking linearly or computationally before this class, and so for me to now be put in a position where I can't just be almost ADD in how I select information and synthesize it, I have to be very formulaic. That was the jump for me, was, "Wow" the precision of language is so important, and it's not just in how we speak, it's how we direct this computer system." So yeah. That for me was the hardest part was, just wrangling my brain into thinking that way.

Carrie and Rebecca realize that CT is different from how they think normally and talk about how they had to shift their thinking to align with a computational perspective. We also saw a similar computational thinking shift with Rebecca in the Computational Concepts section. We asked if they used the CLF poster as a resource, and both agreed. Carrie talks about the CLF poster questions pushing her towards a computational perspective:

Carrie: Having [the poster prompts] asked of me were excellent at leading me towards the answer . . . they were good starting points to get to that place, to get my brain working again.

Here we can see that Carrie tries to work through problems with a computational perspective and uses the CLF poster as a tool to focus her thinking and get her working in the right direction. Carrie is developing her computational thinking and recognizes that the poster is a resource that can help her when she is stuck and can get her back in a computational mindset again.

4.3 Computational Practices

This section outlines three themes we saw in our analysis of computing practices: programming with structure, peer-to-peer collaboration, and the debugging process.

4.3.1 Programming with Structure. Though all of the focus group participants had taken previous classes, their experiences were with scripting (R) and markup (HTML) languages. Caitlyn was used to creating web pages with the very structured and formulaic HTML and CSS languages. She talks about how she is confused when switching to Python, which is a very terse language:

Caitlyn: I'm so used to Web where there is a structure. Wanda's class almost confuses me because I'm like, "Where do we start? What do we type into Jupyter to make it start?" I don't know, for me it's confusing because I don't have structure. What's the beginning? What's the end? Where do I start from there?

Caitlyn was used to the heavy markup of HTML and internalized its unique formatting requirements as a computational practice. When she had to start programming from a blank file when was uncertain about how and where to start coding. When she describes the thought process she goes through when she starts programming, she sounds overwhelmed and keeps asking where to begin. Caitlyn does not seem to have developed her computational thinking enough to be able to clearly and confidently begin programming. Carrie also talks about her experiences with programming in Python for the first time:

Carrie: My first attempt at coding was so precise. You realize that not only can you be precise, but you can be brief in getting your point across to the computer. It's really interesting how it's the synthesis of ideas, but it's also the shortening and precision aspect is also so inherent to it.

Carrie talks about discovering a particular trait of programming: precision; and describes how she adapts and learns how to work with precision. In the Computational Perspective section, Carrie mentions the importance of language precision when she is developing how to think in a computational way. She discovers language precision as she's programming and categorizes it as a computational practice, something that will need to be thought about every

time she writes a program. If we contrast Caitlyn and Carrie's experiences, we can see that both women face challenges when learning Python; however, while Carrie internalized this new challenge as a computing practice and vital to learning programming, Caitlyn is still trying to use computational practices she learned from her previous experience programming. This has different outcomes for both women: Carrie continues to develop her computational thinking while Caitlyn still has an underdeveloped notion of CT.

4.3.2 Peer-to-Peer Collaboration. Carrie and Rebecca worked with another student, Lucy, during every day of the class observations; while Caitlyn always worked alone. We asked Carrie and Rebecca if they used the CLF poster, they responded that they used the CLF poster as a resource that helped their team collaboration and problem solving:

> *Rebecca: So those questions help solidify information, which is so important if you want to carry it. If you want to get your assignment done, or if you want to carry it through classes into the future. Having someone ask you or ask a question that forces you to figure it out too. Not just get an answer. So that's what I really appreciate about this kind of thing.*
>
> *Carrie: But these are really important when you're first learning a new skill. To really question yourself. Question your peers. Have them be questioning you. To kind of keep the flow going, so there is not a big lull in the conversation or in the programming, where you're just like, "I don't know what the hell I'm doing or where I should start."*

Both Carrie and Rebecca view collaboration as a computational practice that helps them retain and better understand the material. They have success using the poster as a tool to help guide their collaborative discussions and provide meaningful support to each other. Carrie talks about how it can be difficult to start or keep a conversation flowing when they have an inexperienced group, but the CLF poster provides a starting place for that conversation to happen. She also talks about being unsure of how to begin programming, the same problem Caitlyn talks also but says the CLF poster helped through this process. Rebecca and Carrie see their group collaboration as an important computational practice and view the CLF poster to facilitate their group workings. When Caitlyn was asked about working with her peers, she states that she doesn't like working through problems with others:

> *Caitlyn: I feel like I can figure it out on my own and when I'm working with someone else I'm like, "Tell me what you think." Then I go away and I think about it on my own. I just feel like sometimes people arrive at answers slower than I do. I want to hear what you have to say, I just don't want to work through this process with you. Just tell me your thoughts and I'll work backwards and figure it out.*

Caitlyn sees working alone as a computational practice. The is the second time Caitlyn talks about preferring to work alone when programming. She does not see the benefit in working with other people and thinks they will slow her down. Caitlyn's method of "working backward" depends on someone always telling her the answer and her being always being able to figure it out. This method doesn't teach her how to analyze her problems or work to build a solution on her own.

4.3.3 The Debugging Process. Finally, we will contrast scenarios where the CLF poster users and the non-CLF poster users worked through problems with differing methods. The first scenarios show the difference in the type of questions that are asked with and without the CLF poster. The second set of scenarios compare how much agency students have when they are being helped.

During the class observations we saw that many students ran into problems when they tried to use strings in their boolean expressions, often they would forget to use quotation marks when using their strings. Sean gives an example of a process he might go through when he helps students with this type of error:

> *Sean: First off I would be like, "Do you want that to be a string?" . . . I don't know, some students didn't quite catch on what is string was. So I'd be like, "Do you want that to be defining text?" They guessed, "That's what I want." Okay, "What makes it different from the rest of the code that you have right now?" Hopefully push them in that direction. Then at the end, if they don't get it, tell them and explain why it needs to be formatted that way.*

When Sean describes his process, it is clear that he is trying to lead the students through their problem and not just give them correct answer; however, two of his example questions are yes or no questions, which do not push students to analyze their problem deeper. His third example question is a leading question that does push students to think about their code and make inferences about what needs to be changed. Sean's methods do not help students develop their computational thinking. Comparatively, Rebecca talks about how her group member Lucy helped her with a similar problem:

> *Rebecca: Lucy asks us, "Hey, what's in that cell that shouldn't be?" Or, "Hey, what's in that string is missing?" Asking those questions, when I find it, I commit that to memory, and it works better.*

Lucy, a CLF-poster user, helps Rebecca by asking her leading questions immediately. Like Sean, Lucy asks Rebecca to compare what she already has in her code to help her find a solution. Lucy lets Rebecca work through the entire problem herself until she finds the correct solution, unlike Sean who tells his students the answer when they are struggling to find a solution.

These next two examples compare how much agency the students needing help has in the debugging process. In the Computational Concepts section, there is an excerpt where Sean helps Caitlyn with her "Choose Your Own Adventure" game. If we focus on Sean's actions during this encounter when he helps Caitlyn, he is seen nearly explicitly telling Caitlyn how to debug her program. Though he is explaining his actions as he goes, he does not let Caitlyn work through her algorithm herself and he prevents her from being able to work out the problem herself. In this interaction, Sean is also leading the discussion and only asks for Caitlyn's answer to a yes or no question at the very end of his explanations. We see both students having trouble with novice level explanations of

computing concepts and Sean dominating the conversation with Caitlyn and telling her and answers instead of helping her learn on her own.

Carrie and Rebecca also brought up experiencing difficulties with their "Choose Your Own Adventure" game, specifically formatting their nested conditional statements. They discuss how Lucy worked with them and went through the CLF poster prompts to help solve their problems:

> *Carrie: I was having a problem with a certain part of the code and we'd gone through it several times and it was still bringing up error messages. [Lucy] was, "I'm not going to tell you. We're just going to go through these [CLF poster] questions until you find it."*
>
> *Rebecca: It's nice to have questions that not only put that person on the spot to tell you, "How did you get that answer, tell me." Creative ways to come up with questions that are like, "How can I also have this best ingrained in my mind too, for how I work and not just copy someone."*

Though Lucy is leading their interactions, in both of the women's examples they control the pacing of the conversation by explaining their problem and how they got to that point. Carrie and Rebecca are prompted to think out loud and reach the solution on their own.

5 DISCUSSION

Though our case studies of Carrie and Rebecca, we can see that the CLF poster helped them develop their computational thinking skills and gave them a format that enabled them to work through their programming problems and created a deeper understanding of the computational concepts. The motivation for this project was to create an intervention tool that provided scaffolding for untrained students to engage in meaningful collaboration. The team of Carrie, Rebecca, and Lucy were novice programmers who did not have experience tutoring their peers. Carrie and Rebecca ran into some initial cognitive dissonance when they tried to think about their programming problems using their normal thinking perspective, however, once they worked on adopting a computational perspective, they were able to see more success when conceptualizing their ideas. All three women worked in a group and saw the benefits of being able to work through the CLF poster prompts with their group members. They talk about how the poster prompts provided them with a way to start collaborative discussions. Finally, we can see how Lucy was able to utilize the CLF poster to help Carrie and Rebecca work through their problems and find solutions on their own.

Caitlyn, a non-CLF poster user showed trouble in describing her programming problems, getting confused between different languageâĂŹs computational practices, conflated learning computational thinking and learning computational concepts. She held on to her non-computational way of thinking and preferred to work alone. She did not develop strong computational thinking skills and did not benefit from the CLF poster.

Sean, the course assistant, did not use the CLF poster and the guidance he offered students did not consistently help them develop computational thinking skills or better understand the courseâĂŹs computational concepts. He was seen having trouble describing programming concepts with the correct terminology, helping students in a way that did not foster deeper thinking about the problems and being overbearing in his interactions with students who needed help.

6 FUTURE WORK

This study provided us with evidence to the CLF posterâĂŹs usefulness for students who adopt a computational perspective and use this resource as a computational practice; however, we are still unsure if it can be a useful tool for students who do not develop computational thinking skills. Future work will involve a poster redesign and find ways to encourage poster use for additional students.

REFERENCES

[1] L. J. Barker and K. Garvin-Doxas. 2004. Making Visible the Behaviors that Influence Learning Environment: A Qualitative Exploration of Computer Science Classrooms. *Computer Science Education* 14, 2 (2004), 119–145.

[2] K. Brennan and M. Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (AERA '12)*. Vancouver, Canada.

[3] H. Danielsiek, J. Vahrenhold, P. Hubwieser, J. Krugel, J. Magenheim, L. Ohrndorf, d. Ossenschmidt, and N. Schaper. 2017. Undergraduate teaching assistants in computer science: Teaching-related beliefs, tasks, and competences. In *2017 IEEE Global Engineering Education Conference (EDUCON '17)*. Athens, Greece.

[4] A. Decker, P. Ventura, and C. Egert. 2006. Through the Looking Glass: Reflections on Using Undergraduate Teaching Assistants in CS1. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. New York, NY, USA.

[5] L. Fingerson and A. B. Culley. 2001. Collaborators in Teaching and Learning: Undergraduate Teaching Assistants in the Classroom. *Teaching Sociology* 29, 3 (2001), 299–315.

[6] J. Forbes, D. J. Malan, H. Pon-Barry, S. Reges, and M. Sahami. 2017. Scaling Introductory Courses Using Undergraduate Teaching Assistants. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. New York, NY, USA.

[7] C. Gregg and C. M. Lewis. 2015. Working with Undergraduate Teaching Assistants: Best Practices and Lessons Learned. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. New York, NY, USA.

[8] D. G. Kay. 1998. Large Introductory Computer Science Classes: Strategies for Effective Course Management. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '98)*. New York, NY, USA.

[9] E. Mercier, C. Fong, R. Cober, J. D. Slotta, K. S. Forssell, M. Isreal, A Joyce-Gibbons, and N. Rummel. 2015. Researching and Designing for the Orchestration of Learning in the CSCL Classroom. In *Exploring the Material Conditions of Learning: The Computer Supported Collaborative Learning (CSCL) Conference (CSCL '15)*. Gothenburg, Sweden.

[10] M. Park and T. Lash. 2015. Collaborative Discussion Framework. (2015). Retrieved 2017-06-13 from http://ctrlshift.mste.illinois.edu/2015/04/03/collaborative-discussion-framework/

[11] J. Parker, R. Cupper, C. Kelemen, D. Molnar, and G. Scragg. 1990. Laboratories in the Computer Science Curriculum. *Computer Science Education* 1, 3 (1990), 205–221.

[12] C. Stewart-Gardiner, D. G. Kay, J. C. Little, J. D. Chase, J. Fendrich, L. A. Williams, and U. Wolz. 2001. Collaboration vs Plagiarism in Computer Science Programming Courses. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*. New York, NY, USA.

[13] N. Titterton, C. M. Lewis, and M. Clancy. 2010. Experiences with lab-centric instruction. *Computer Science Education* 20, 2 (2010), 79–102.

[14] J. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (2006), 33–35.