A Review of Cognitive Apprenticeship Methods in Computing Education Research

Anshul Shah
ayshah@ucsd.edu
University of California, San Diego
USA

Adalbert Gerald Soosai Raj
asoosairaj@ucsd.edu
University of California, San Diego
USA

ABSTRACT

Cognitive Apprenticeship (CA) is an instructional model that outlines how experts can transfer their skills and knowledge to a learner for reasoning-based tasks, such as reading comprehension or mathematical problem solving. Specifically, CA includes 6 teaching methods—modeling, scaffolding, coaching, reflection, articulation, and exploration—that facilitate learners' observation, acquisition, and externalization of implicit processes and techniques for completing a task. In this paper, we present a systematic literature review of 143 conference papers across ACM and IEEE venues about CA in computer science education literature. Specifically, we aim to understand which teaching methods are typically referenced, the theory level (i.e., depth of CA theory discussion) present in the literature, and the key findings related to CA-based teaching approaches. Our review reveals that CA has been cited in computing education research as a guiding theory for various course designs, though there is a clear emphasis on papers related to modeling, scaffolding, and coaching whereas reflection, articulation, and exploration are under-explored. We found that CA methods have been effective in improving students' enthusiasm towards computing, improving pass-rates in courses, and improving instructors' capacity to accommodate more students by reducing instructor workload. However, a key challenge of CA approaches that emerged from our review is the difficulty in scaling the approach in settings with a high student to instructor ratio. Through this literature review, we aim to highlight effective CA approaches and how future initiatives can leverage CA to improve student learning.

CCS CONCEPTS

• **Social and professional topics → Model curricula**.

KEYWORDS

Cognitive Apprenticeship, teaching methods, active learning, learning environments, managing enrollment growth

ACM Reference Format:
Anshul Shah and Adalbert Gerald Soosai Raj. 2024. A Review of Cognitive Apprenticeship Methods in Computing Education Research. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630769

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630769

1 INTRODUCTION

Apprenticeship is one of the oldest forms of knowledge transfer in human history. Dating back thousands of years, experts in certain domains, such as metallurgy and blacksmithing, would teach their craft to an apprentice through observation and guided practice. This approach was and is an effective way for learners to develop vocational skills. Today, however, this apprenticeship model has largely been replaced by the modern schooling system that aims to prepare students for a number of potential careers. As a result, the Cognitive Apprenticeship (CA) model was proposed in an effort to bring the traditional apprenticeship model into the classroom [7]. Introduced in 1991, CA provides a model for how experts in a complex craft can impart their expertise to learners, such as their domain knowledge, problem-solving techniques, and learning strategies, by making their thinking visible to the learner [7].

Though CA was introduced in the context of K-12 reading, writing, and math education, CA approaches have been applied in a range of disciplines and at varying education levels [11]. Decades of research suggests that the CA model is an accurate representation of how learning generally occurs [11]. Therefore, the goal of this paper is to understand the current state of research surrounding CA teaching methods as it relates to teaching and learning computing. Specifically, we aim to understand the extent to which CA methods are used in computing education research literature and the empirical impacts of CA approaches. Given previous recommendations to employ theory as a guide for educational initiatives [2, 11], this systematic literature review sheds light on how CA has been used in computing education literature and reveals avenues for future work to explore and implement effective CA approaches.

2 BACKGROUND AND RELATED WORK

Cognitive Apprenticeship is an educational theory proposed by Collins et al. in 1991 that encompasses four *dimensions*—Content, Methods, Sequencing, and Sociology [7]. These dimensions are interconnected and vital for a learning environment that facilitates the transfer of expertise from expert to learner [7]. The framework directs instructors to create well-designed tasks (i.e., Sequencing) and to use effective teaching strategies (i.e., Methods) to help learners gain control of heuristic, learning, and control strategies (i.e., Content). By allowing students to collaborate with peers in a community of practice (i.e., Sociology), experts can situate students' learning in an authentic environment [7].

CA is a broad theory. Each dimension alone can be the subject of dozens of research studies (e.g. "community of practice"—part of the Sociology dimension—has almost 7,000 results in the ACM Digital Library after filtering for SIGCSE-sponsored venues). Therefore, we limit the scope of this literature review to the ways that CA

methods have been used in computing education literature. Each of the following teaching methods accomplishes a specific phase in the transfer of expertise from expert to learner:

(1) *Modeling* involves an instructor demonstrating the process of completing a task while verbalizing their approach.
(2) *Scaffolding* involves an instructor providing tasks of appropriate difficulty and scope for learners to practice their skills. Part of the scaffolding method involves *fading*, in which an instructor slowly reduces the scaffolds for learners to complete tasks independently.
(3) *Coaching* involves an instructor providing feedback as learners complete tasks.
(4) *Reflection* involves the learner thinking about the effectiveness of their techniques, comparing their process to the instructor, or using other self-evaluation practices.
(5) *Articulation* involves the learners verbalizing their reasoning, justifying their approach, or explaining their knowledge as they complete tasks.
(6) *Exploration* involves the learner conducting open-ended tasks with minimal to no involvement from the instructor. The exploration method can be achieved through the fading of instructor scaffolds.

Modeling, scaffolding, and coaching are described as the "core" of CA, as they also exist in traditional apprenticeship [7]. These three methods are essential for a learner to acquire skills via observation and guided practice [7]. By contrast, the next three methods are added to the traditional apprenticeship model to facilitate the transfer of expertise for reasoning-based tasks—the key difference between a traditional and a cognitive apprenticeship. In fact, Collins et al. note that the reflection and articulation methods are necessary for students to "gain conscious access to and control of" their problem solving process while exploration helps develop learner autonomy in applying the expert skills and processes they've learned [7]. Therefore, all six CA methods are vital for facilitating the transfer of domain knowledge, heuristic strategies, learning strategies, and control strategies to learners.

As a result, a guiding motivation in our work is to synthesize the research surrounding the CA methods in computing education. To do this, we adopt a similar motivation and approach as Minshew et al., who conducted a literature review of CA in graduate STEM education to understand which CA dimensions (methods, content, sequencing, sociology) were emphasized in graduate STEM education and common ways CA is implemented in graduate programs [25]. In their review, Minshew et al. found that the literature surrounding graduate STEM education focused far more on scaffolding and coaching rather than articulation and reflection [25]. The findings also revealed a lack of work on the Sequencing dimension compared to the Methods and Content dimensions [25]. Though we have a more narrow focus in this paper (we focus only on the CA methods while Minshew et al. focus on all 4 dimensions), we employ a similar search and analysis approach to Minshew et al..

In 2008, Dennen and Burner conducted an extensive review of CA in educational practice. The review showed that CA approaches have been used in fields such as teacher education, doctoral programs, nursing, and engineering with empirical success [11, 25]. The review identified approaches such as mentoring, scaffolding, and situated learning across disciplines at various educational levels and concluded that CA is an accurate representation of the learning process [11]. However, one of the final recommendations by Dennen and Burner was for a more systematic program of studies aimed at developing guidelines for effective implementations of CA in teaching and learning [11]. To our knowledge, no work has aimed to synthesize the studies regarding CA in computing education. Therefore, a contribution of our present work is to synthesize the literature surrounding CA methods specifically within computing education so that our community can understand the impacts of these methods and design effective CA approaches.

3 RESEARCH QUESTIONS

(1) Which Cognitive Apprenticeship Methods are emphasized in peer-reviewed studies in computing education research?
(2) To what degree (i.e., theory level) is Cognitive Apprenticeship discussed in such studies?
(3) What benefits and challenges are present in literature that tests a Cognitive Apprenticeship approach to teaching programming?

4 METHODS

4.1 Search Methods

We followed an extremely similar search method as previous theory-related literature reviews, such as a prior review of research related to metacognition by Loksa et al. [23] and a review of CA in STEM education by Minshew et al. [25]. Because our literature review aims to understand the use of CA within computing education, we searched the ACM Digital Library (DL) and the IEEE Xplore archive for papers that related to CA.

4.1.1 ACM Digital Library Search. Within the ACM DL, we conducted an advanced search for the phrase "cognitive apprenticeship" within the full paper text. We chose to search the full paper text rather than only the abstract or title because we wanted to understand the *theory level* (i.e., depth of theory discussion) of the papers that refer to Cognitive Apprenticeship. Therefore, we did not want to limit our corpus to papers that *only* mention CA in the title or abstract since this criteria would exclude papers that mention CA in the related work or discussion sections.

Unlike searching for a topic with multiple variations (such as "metacognitive", "metacognition", "self-regulatory", "self-regulation", etc.), we felt that cognitive apprenticeship is a straight-forward term and theory to include without needing other variations. From a quick manual inspection, broadening the search to only "apprentice" or "apprenticeship" included many papers that mention the terms colloquially rather than as a reference to the Cognitive Apprenticeship theory. Therefore, we proceeded with only searching for "cognitive apprenticeship".

We then applied a filter for SIGCSE-sponsored venues, which returned all papers at the SIGCSE Technical Symposia, ITiCSE, ACE, and ICER. In total, 63 papers were returned from this search criteria. However, we noted that the Koli Calling and TOCE venues were not included in the SIGCSE-sponsored venues, so we altered the filters to include the Koli Colling conference under "Proceedings Series" and the TOCE under "Journal/Magazine Names." With these additional filters, we found 9 papers from Koli Calling and 13 from

TOCE, bringing our total papers from the ACM DL search to 85. The most recent time this ACM DL Search was run and confirmed was on August 16, 2023.

4.1.2 IEEE Xplore Search. Similar to the ACM DL search, we conducted an advanced search on the IEEE Xplore Archive for the phrase "cognitive apprenticeship" within the full paper text. We applied the following three filters to our advanced search: "computer science education", "Journals", and "Conferences". Notably, the "computer science education" filter was necessary because of the prevalence of papers that mention CA outside of a computing education context. This filter does not eliminate papers based on whether the *venue* is within the "computer science education" domain but rather if the content of the paper is related to "computer science education." In total, we found 76 papers from this search on IEEE Xplore. The most recent time this IEEE Xplore Search was run and confirmed was on August 16, 2023.

4.2 Analysis Methods

In total, we had 161 papers to analyze—85 from the ACM DL search and 76 from the IEEE Xplore search. The first author checked each paper against our exclusion criteria and reviewed the content according to our data collection categories. Given our limitation that only one author analyzed the corpus, we designed our exclusion and categorization criteria to be sufficiently clear and well-defined such that subjectivity in the analysis was minimized. Therefore, the theory level and CA methods of the papers are based on the explicit mention of specific keywords located in the paper. To further mitigate this concern, the first author independently reviewed the corpus twice without referring to the first round of categorizations.

4.2.1 Exclusion Criteria. We removed papers that:

(1) did not mention "Cognitive Apprenticeship" or any similar variant ("cognitively apprenticed", "apprenticeship", "apprentice", etc.).
(2) were not conference or journal papers.
(3) were deemed outside the realm of computing education.

We removed 18 papers based on this criteria. Of the 18 papers we removed, 12 did not mention "Cognitive Apprenticeship" (all from IEEE Xplore), 5 were not conference or journal papers (all from the ACM DL), and 1 was about simulation apprenticeships in industry. We were left with 143 total papers in our corpus.

4.2.2 Data Collection Categories. We categorized the papers in our corpus on two factors based on how each paper referenced CA.

- **Theory level:** Adapted from Kumasi et al. [21], the theory level represents the extent to which authors mention a theory in a paper. We categorize a paper as one of *theory dropping*, *theory relating*, *theory application*, *theory testing*, and *theory generating*, which are each described in Table 1.
- **CA methods:** We note down any of the CA methods—modeling, scaffolding, coaching, reflection, articulation, and exploration—that are *explicitly mentioned* in the paper.

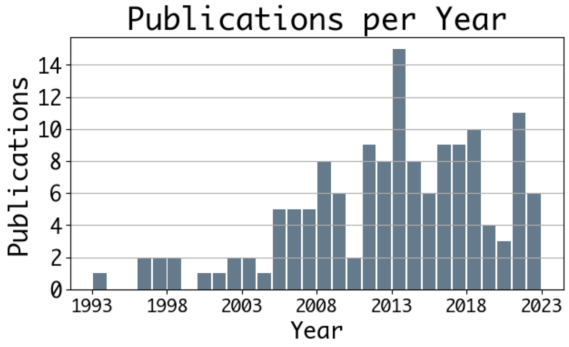
5 RESULTS

We analyzed and classified 143 total papers. The list of all 143 papers and their categorizations can be found at: https://bit.ly/ca-literature. As seen in Figure 1, the number of papers mentioning

Table 1: Theory levels considered during literature review, from Kumasi et al. [21] and Minshew et al. [25].

Label	Description
Theory Generating	Building upon or revising the Cognitive Apprenticeship theory to create a new theory.
Theory Testing	Describing an approach based on Cognitive Apprenticeship *and* evaluating student outcomes.
Theory Application	Designing an approach based on Cognitive Apprenticeship *without* evaluating the approach, employing theory throughout the paper.
Theory Relating	Mentioning Cognitive Apprenticeship in the discussion to connect the study to theory.
Theory Dropping	Mentioning Cognitive Apprenticeship *only* in the related work or background and not revisited later.

Figure 1: Publication years in our corpus (n = 143).



CA in computing education was relatively low in the 1990s and early 2000s. Though Collins et al. initially proposed the Cognitive Apprenticeship theory in 1991 [7], we suspect that references to CA theory increased after 2005 when the Cambridge Handbook of the Learning Sciences, which includes a chapter on CA, was released [1].

5.1 RQ1: Frequency of CA Methods

Of the 143 papers, 70 (49%) did not explicitly mention any CA methods. In these 70 papers, we often found references to other key tenets of CA, such as the emphasis on process-oriented skills [24, 32, 36] or the importance of situated learning and cognition [6, 10, 33]. Typically, these papers that did not mention any methods also did not refer to theory much. In fact, 55 of the 70 papers (78.6%) were *theory dropping* papers, meaning that CA was mentioned briefly in the related work or background section.

On the other hand, 73 of the 143 papers (51%) explicitly mentioned at least one method. Table 2 shows the frequency of explicit references to each method in our corpus. Each paper can cite multiple methods; therefore, the frequencies do not sum to 143. Scaffolding was the most commonly-mentioned method we encountered, followed by modeling and coaching. We noticed a clear dichotomy between the first three methods—modeling, scaffolding, and coaching—receiving many more references than the last three

Table 2: CA Methods mentioned in our corpus (n=143). A single paper can mention as few as 0 methods or all 6 methods.

Label	Frequency
Modeling	42 (29.4%)
Scaffolding	61 (42.7%)
Coaching	38 (26.6%)
Reflection	22 (15.4%)
Articulation	17 (11.9%)
Exploration	10 (7%)

Table 3: Theory levels present in corpus (n=143).

Theory Level	Frequency
Theory Generating	0 (0%)
Theory Testing	17 (11.9%)
Theory Application	21 (14.7%)
Theory Relating	30 (20.1%)
Theory Dropping	77 (53.8%)

methods—reflection, articulation, and exploration. In fact, of the 73 papers that mention any CA methods 70 (95.9%) papers refer to at least one of the first three methods whereas only 23 (31.5%) refer to one of the last three methods.

5.2 RQ2: Theory Levels in CA Literature

Table 3 shows the frequency of each theory level in our corpus. Only 17 papers (11.9%) fell into the *theory testing* category, which means it evaluates a pedagogical technique, course design, or tool that was design using a CA approach on students learning, attitudes, or some other outcome factor.

Overall, a strong majority of the papers in our corpus were *theory dropping* or *theory relating*, which are considered *minimal* levels of theory [21, 25]. In fact, over half (53.8%) of the papers were *theory dropping*, meaning that CA was very briefly mentioned in an early section of the paper without being revisited later in the methods or discussion sections. We also found 21 papers that were *theory application*, meaning that the authors presented an approach based on CA without a meaningful evaluation of the approach. Notably, we did not discover any *theory generating* papers that cited CA.

5.3 RQ3: Key Benefits and Challenges of CA Approaches

In this section, we identify the key benefits and challenges found in the 17 *theory testing* papers, as listed in Table 4. While each of these studies was motivated by a CA approach, the specific approaches vary greatly among the papers. For example, only three papers covered all 6 methods in their course design [4, 16, 22]. Conversely, the papers that discuss Xtreme Apprenticeship—a CS1 course designed by researchers in Finland where students complete programming activities during lecture while getting help from instructors—only makes references to modeling, scaffolding, and coaching [19, 37–39]. Table 5 displays only the frequencies of the Methods mentioned in the 17 *theory testing* papers. Again, we notice

Table 4: Benefits and challenges identified in *theory testing* papers in our corpus.

Label	Description
Benefit: Student Enthusiasm	[12], [18], [19], [22], [34], [35], [39]
Benefit: Pass Rates and Performance	[4], [9], [13], [19], [20], [22], [27], [37], [38]
Benefit: Managing Enrollment Growth	[3], [14], [27], [35], [37], [39]
Challenge: Scalability	[4], [20], [37]

Table 5: CA methods among *theory testing* papers (n = 17).

Label	Frequency
Modeling	14 (82.4%)
Scaffolding	16 (94.1%)
Coaching	13 (76.5%)
Reflection	5 (29.4%)
Articulation	4 (23.5%)
Exploration	3 (17.6%)

the same trend that was present in the overall corpus—a prevalence of papers related to modeling, scaffolding, and coaching and a lack of papers about reflection, articulation, and exploration.

5.3.1 Benefit: Enthusiasm. One benefit that emerged from the *theory testing* literature was improved student enthusiasm for computing after completing courses designed with CA methods [18, 19, 27, 39]. These courses typically included a "hands-on," active learning component, such as coding exercises completed in Xtreme Apprenticeship [37] or tangible robotics projects [22]. Outcomes we found included an increase in student interest in science [22] and programming [34], greater commitment to a software engineering career [35], an increase in programming courses taken by students within the next year [19], and an increase in "eager" teaching assistants following an Xtreme Apprenticeship course [39].

5.3.2 Benefit: Pass-Rates and Performance. Though a relationship likely exists between greater student enthusiasm and improved performance, we identified a set of studies that demonstrate specific improvements in course pass rates and student abilities after a CA teaching approach. A wealth of data shows the significant impact of Xtreme Apprenticeship on pass rates in both introductory and advanced programming courses [19, 37, 38]. Specifically, Keijonen et al. found that even 7 to 13 months *after* students completed an Xtreme Apprenticeship course, "grade distribution, pass-rate, overall credit accumulation, and student success in staying on the desired study path have all improved" [19]. Knobelsdorf et al. saw similar improvements in pass-rates in a theoretical computer science course after including specific CA methods of modeling, scaffolding, and coaching [20]. Finally, Jin and Corbett showed that a CA curriculum that emphasized the problem-solving process and provided coaching to students led to a 48% gain on student post-tests compared to a non-CA approach that involved little to no coaching [18].

5.3.3 Benefit: Managing Enrollment Growth. CA approaches also helped instructors manage larger classes by reducing students' help requests. For example, Murray et al. developed an online tool to provide students with prompts and hints that faded over time and noticed a 65% reduction in scaffold requests, which the authors attribute to "increas[ed] self-reliance" [27]. Similarly, Upchurch and Sims-Knight noted that students' spent significantly more time reviewing their own code and reported a greater tendency to create improvement plans after being introduced to a peer code review activity [35]. The impact of a reduction in help requests was most clearly demonstrated by Feldgen et al., who introduced scaffolds such as demonstration videos, peer reviews, and self-reflection questions in their distributed systems course [14]. The authors note that despite the class size increasing from 6 students to 20 students, the increased enrollment "did not represent an increase in the extra support time" due to the scaffolds that were introduced [14]. In a similar vein, one study about Xtreme Apprenticeship showed that the CA approach can increase the number of "eager" and capable teaching assistants (perhaps due to improved student enthusiasm) to help administer the course they just completed [39].

5.3.4 Challenge: Difficulty in Scaling CA Approaches. Conversely, several studies acknowledged the difficulty in scaling a CA approach. For example, Knobelsdorf et al., who taught a CA-motivated theoretical computer science class, share that they could not cover all aspects of CA because of the high teacher-learner-ratio and limited number of study sessions [20]. Similarly, Armarego share their experience of attempting to scaffold and fade instructor support but encountered a group of students that were unhappy with the fading and demanded more assistance [3]. Finally, though Bareiss and Radley were able to provide an effective software engineering course covering all 6 CA methods, the authors acknowledge the high time commitment for faculty time (some faculty reported spending over one and a half hours for each student in the course *per week* to provide effective coaching) [4].

6 DISCUSSION

6.1 Emphasis on Modeling, Scaffolding, and Coaching

We hypothesize several reasons for the emphasis on modeling, scaffolding, and coaching over reflection, articulation, and exploration—a trend that was also present in Minshew et al.'s review of CA in graduate STEM education [25]. First, Collins et al. mention that modeling, scaffolding, and coaching make up the "core" of CA, which may have led to other authors disregarding the other methods as less-important, peripheral methods. In fact, we discovered some papers that aimed to implement a CA model but *only* made reference to modeling, scaffolding, and coaching without even acknowledging the remaining three phases [3, 13, 18, 34], including the line of work on Xtreme Apprenticeship [19, 37, 38]. Based on the text of the papers, it is unclear if these interventions decided against using these methods or were unaware of them. A second potential reason for the lack of emphasis on reflection, articulation, and exploration is the difficulty in using such methods for a large class size. After their CA-motivated course redesign, Knobelsdorf et al. acknowledged that while it is important for students to articulate and reflect, they could only use the modeling and scaffolding methods because of the large class size and limited meeting times [20]. We reason that reviewing and assessing student reflections, open-ended articulations, or exploratory assignments can be a time-consuming process for the instructional staff, thereby limiting the number of papers that evaluate these methods.

The lack of evaluative work on the reflection, articulation, and exploration methods may be problematic from a pedagogical perspective. Collins et al. point out that the last three methods are necessary steps for learners to gain control of the strategies they learned in the first three methods and to apply these strategies in an open-ended setting [7]. Therefore, future work may investigate lightweight interventions that engage reflection, articulation, and exploration methods. Specifically, we hypothesize that CA approaches that focus on the last three methods may help address the academia-industry gap, since some papers have pointed to student struggles with communication [29] (which may be addressed through an emphasis on articulation) and working independently on pre-existing code bases [5] (which may be addressed through an emphasis on exploration).

Although we discovered an under-utilization of reflection, articulation, and exploration in the CA literature, we caution that these findings do not indicate an under-emphasis of these approaches within the broader computing education literature. For example, plenty of work has been conducted on reflection through a metacognition lens [23], such as work on exam wrappers [8]. Nonetheless, we encourage future work to target reflection, articulation, and exploration through a CA lens in order to adequately evaluate the learning theory in a computing context.

6.2 Authenticity and Active Learning in CA Approaches

Providing an authentic learning environment to real-world situations is a core tenet of CA [7] and was a consistent theme among *theory testing* papers [4, 22, 35]. Many of the CA approaches included hands-on activities (scaffolding) and frequent feedback from instructors (coaching), which typically involved a form of active learning to provide an "authentic" programming or engineering experience for students [4, 13, 20, 22, 34, 35, 38]. Examples include students programming tasks for robots to complete [22], using a hands-on prototyping platform to work with tangible devices [34], and participating in the lab-centered, Xtreme Apprenticeship approach [19, 39].

Authenticity and active learning may contribute to the benefits that we observed in the literature. For example, we found that approaches that used an authentic learning environment in which students collaborated and applied their knowledge generated student enthusiasm for the course content and a increased students' self-reported interest in pursuing a computing career [4, 22, 35]. Of course, these findings are not new for the computing education field. For example, prior work has pointed to numerous learning benefits of active learning techniques, including greater retention [28]. However, our findings demonstrate how these instructional techniques can effectively be part of a CA-motivated course design. Therefore, we encourage future CA interventions to apply the CA

methods in authentic, "learning-by-doing" environments to realize benefits such as improved pass-rates and student performance.

6.3 Using CA to Manage Enrollment Growth

The competing themes of managing enrollment growth and difficulty in scaling a CA approach were apparent in our findings. Since a CA approach, specifically through scaffolding and fading, aims to deliberately reduce student dependence on the instructor by removing scaffolds over time, a successful CA approach could reduce student help requests of the instructor. Indeed, research from Murray et al. and Feldgen et al. empirically demonstrated this reduction in student help requests and improvement in students' self-reliance [14, 27]. However, despite these benefits, we note that there certainly exists an initial "start-up" cost to designing, developing, and implementing such scaffolds. Fortunately, these papers demonstrated a long-term, improved capacity for managing more students over several years of offering the same course [14], especially in the case of Xtreme Apprenticeship [19, 39].

On the other hand, some papers outlined the significant time and resources needed to provide an effective coaching experience to students, especially in upper-division courses [4, 20]. We suspect that topics such as advanced software engineering [4] or theoretical computer science [20] require students to develop more complex heuristic and control strategies, thereby relying on more instructor guidance and feedback than in introductory courses. Indeed, the original CA work by Collins et al. was aimed at K-12 education for skills such as reading comprehension, mathematical problem-solving, and writing [7]. The differences between a K-12 environment and a university setting may explain the difficulty in scaling a CA approach, as universities typically have a higher ratio of students to instructors with less meeting times.

Our findings indicate that when scaffolding and fading are done effectively, instructors have reported a long-term, improved capacity to teach more students. The challenge for instructors seems to be with striking a balance between sufficient help for students early in a course and deliberately reducing the assistance they provide to facilitate students' independence. For example, designing an open-ended, slightly-ambiguous task for students to complete would leverage the *exploration* method, but may open the door to student confusion and an uptick in help requests due to the ambiguity of the assignment. Therefore, we note that targeted interventions with the goal of improving students' self-reliance so that less instructor support is needed may be a viable approach to use CA to manage enrollment growth.

Notably, a sub-theme we detected in our analysis was the use of online tools and learning platforms to provide personalized scaffolding and coaching. Though we only saw two *theory testing* work that presented an online tool that leveraged CA methods [18, 27], we envision an avenue of future work that aims to scale a CA approach through intelligent tutoring systems and other personalized tools. Though the recommendation of using computer-mediated environments for CA is not new [11], programming in IDEs provides unique opportunities to facilitate personalized feedback for students. For example, the idea presented by Hundhausen et al. to integrate features in IDEs that use CA methods may be a promising way to provide programming feedback to students at scale while also encouraging reflection and articulation [17]. We encourage future computer-mediated interventions to consider ways to incorporate CA methods—especially reflection, articulation, and exploration—into the design.

7 LIMITATIONS

First, our inclusion and categorization criteria were applied by a single reviewer. We note that several other literature reviews included two reviewers categorizing papers and reporting an inter-rater reliability score for the agreement [23, 25] or have an initial round of deliberation between multiple reviewers and then divide the work [15]. While we did discover reviews that included a single reviewer only [30, 31], we were concerned that only one author applied the criteria to all 143 papers. However, we took steps to limit this concern by aiming to categorize the papers according to relatively objective factors, such as the location of specific keywords. For example, the review by Minshew et al. reported a 100% agreement between two reviewers for deciding the theory level using the same labelling criteria we used in our process [25].

Second, we did not include papers from sources such as SpringerLink, Taylor & Francis, and Google Scholar. We intentionally chose ACM DL and IEEE Xplore as the past literature reviews in computing education that only relied on these venues [26, 30]. Nonetheless, there certainly exist papers that refer to CA in a computing education context outside of ACM and IEEE venues.

Finally, our aims of understanding the benefits and challenges of CA approaches is limited by the explicit connections papers make to CA. Though some studies may have utilized approaches that align with CA methods, the authors may not have included a discussion of CA in the paper. For example, we believe that approaches such as pair programming could be framed as a CA approach that leverages the reflection method (similar to how peer code review has been connected to reflection) and articulation (since students have to explicate their thought process). Nonetheless, a large body of pair programming typically does not refer to CA. As a result, readers should note that our work only represents the course designs and interventions that were explicitly motivated by a CA approach.

8 CONCLUSION

This work demonstrated the prevalence of papers that evaluated modeling, scaffolding, and coaching methods but a lack of papers that evaluated reflection, articulation, and exploration methods. Despite this imbalance of coverage in the CA literature, we identified three benefits of CA approaches from the set of *theory testing* papers: 1) improved student enthusiasm and intention to pursue a computing career, 2) improved pass-rates and student performance, and 3) improved capacity for managing enrollment growth. Conversely, an emerging challenge in implementing CA methods is scaling the approach to accommodate more students while limiting instructor workload. We have identified avenues of future work, such as investigating into the latter-half of the CA methods and approaches for scaling a CA course design to accommodate a higher student-to-instructor ratio.

ACKNOWLEDGMENTS

This work was supported in part by NSF award 2044473.

REFERENCES

[1] 2005. *The Cambridge Handbook of the Learning Sciences.* Cambridge University Press. https://doi.org/10.1017/CBO9780511816833

[2] 2019. *The Cambridge Handbook of Computing Education Research.* Cambridge University Press. https://doi.org/10.1017/9781108654555

[3] Jocelyn Armarego. 2009. Displacing the Sage on the Stage: Student Control of Learning. In *2009 22nd Conference on Software Engineering Education and Training*. 198–201. https://doi.org/10.1109/CSEET.2009.43

[4] Ray Bareiss and Martin Radley. 2010. Coaching via Cognitive Apprenticeship. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 162–166. https://doi.org/10.1145/1734263.1734319

[5] Andrew Begel and Beth Simon. 2008. Struggles of New College Graduates in Their First Software Development Job. *SIGCSE Bull.* 40, 1 (mar 2008), 226–230. https://doi.org/10.1145/1352322.1352218

[6] Jennifer Burg, V. Paul Pauca, et al. 2016. A STEM Incubator to Engage Students in Hands-on, Relevant Learning: A Report from the Field. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (Arequipa, Peru) *(ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 142–147. https://doi.org/10.1145/2899415.2899461

[7] Allan Collins, John Seely Brown, Ann Holum, et al. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[8] Michelle Craig et al. 2016. Introducing and Evaluating Exam Wrappers in CS2. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 285–290. https://doi.org/10.1145/2839509.2844561

[9] Timothy Davis, Robert Geist, Sarah Matzko, and James Westall. 2007. TEXVN: Trial Phase for the New Curriculum. *SIGCSE Bull.* 39, 1 (mar 2007), 415–419. https://doi.org/10.1145/1227504.1227455

[10] Adrienne Decker and David Simkins. 2016. Uncovering difficulties in learning for the intermediate programmer. In *2016 IEEE Frontiers in Education Conference (FIE)*. 1–8. https://doi.org/10.1109/FIE.2016.7757446

[11] Vanessa Dennen and Kerry Burner. 2008. The Cognitive Apprenticeship Model in Educational Practice. *Handbook of Research on Educational Communications and Technology* (01 2008).

[12] Andrew T. Duchowski et al. 2011. TEXNH Trees: A New Course in Data Structures. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 341–346. https://doi.org/10.1145/1953163.1953267

[13] Matthias Ehlenz, Thiemo Leonhardt, et al. 2018. The Lone Wolf Dies, the Pack Survives? Analyzing a Computer Science Learning Application on a Multitouch-Tabletop. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '18)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3279720.3279724

[14] Maria Feldgen et al. 2012. Promoting design skills in distributed systems. In *2012 Frontiers in Education Conference Proceedings*. 1–6. https://doi.org/10.1109/FIE.2012.6462229

[15] Gregor Große-Bölting, Dietrich Gerstenberger, et al. 2023. Identity in Higher Computer Education Research: A Systematic Literature Review. *ACM Trans. Comput. Educ.* (jun 2023). https://doi.org/10.1145/3606707 Just Accepted.

[16] Shen-Tzay Huang, Yi-Pei Cho, and Yu-Jen Lin. 2006. Implementation and Evaluation of Teaching an Introductory Software Engineering Course Framed in Cognitive Apprenticeship. In *2006 13th Asia Pacific Software Engineering Conference (APSEC'06)*. 477–484. https://doi.org/10.1109/APSEC.2006.39

[17] C. D. Hundhausen, D. M. Olivares, and A. S. Carter. 2017. IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda. *ACM Trans. Comput. Educ.* 17, 3, Article 11 (aug 2017), 26 pages. https://doi.org/10.1145/3105759

[18] Wei Jin and Albert Corbett. 2011. Effectiveness of Cognitive Apprenticeship Learning (CAL) and Cognitive Tutors (CT) for Problem Solving Using Fundamental Programming Concepts. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 305–310. https://doi.org/10.1145/1953163.1953254

[19] Hansi Keijonen, Jaakko Kurhila, and Arto Vihavainen. 2013. Carry-on effect in extreme apprenticeship. In *2013 IEEE Frontiers in Education Conference (FIE)*. 1150–1155. https://doi.org/10.1109/FIE.2013.6685011

[20] Maria Knobelsdorf, Christoph Kreitz, and Sebastian Böhne. 2014. Teaching Theoretical Computer Science Using a Cognitive Apprenticeship Approach. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 67–72. https://doi.org/10.1145/2538862.2538944

[21] Kafi D. Kumasi, Deborah H. Charbonneau, et al. 2013. Theory talk in the library science scholarly literature: An exploratory analysis. *Library & Information Science Research* 35, 3 (2013), 175–180. https://doi.org/10.1016/j.lisr.2013.02.004

[22] D. Brian Larkins, J. Christopher Moore, et al. 2013. Application of the Cognitive Apprenticeship Framework to a Middle School Robotics Camp. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 89–94. https://doi.org/10.1145/2445196.2445226

[23] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, et al. 2022. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.* 22, 4, Article 39 (sep 2022), 31 pages. https://doi.org/10.1145/3487050

[24] Mark Mahoney. 2023. Storyteller: Guiding Students Through Code Examples. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1131–1135. https://doi.org/10.1145/3545945.3569843

[25] Lana M. Minshew, Amanda A. Olsen, and Jacqueline E. McLaughlin. 2021. Cognitive Apprenticeship in STEM Graduate Education: A Qualitative Review of the Literature. *AERA Open* 7 (2021). https://doi.org/10.1177/23328584211052044

[26] Diba Mirza, Phillip T. Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 31–40. https://doi.org/10.1145/3291279.3339422

[27] S. Murray et al. 2003. A tool-mediated cognitive apprenticeship approach for a computer engineering course. In *Proceedings 3rd IEEE International Conference on Advanced Technologies*. 2–6. https://doi.org/10.1109/ICALT.2003.1215014

[28] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving Fail Rates Using Peer Instruction: A Study of Four Computer Science Courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 177–182. https://doi.org/10.1145/2445196.2445250

[29] Alex Radermacher and Gursimran Walia. 2013. Gaps between Industry Expectations and the Abilities of Graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 525–530. https://doi.org/10.1145/2445196.2445351

[30] Ana Selvaraj, Eda Zhang, Leo Porter, and Adalbert Gerald Soosai Raj. 2021. Live Coding: A Review of the Literature. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Virtual Event, Germany) *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 164–170. https://doi.org/10.1145/3430665.3456382

[31] Umar Shehzad, Mimi Recker, and Jody Clarke-Midura. 2023. A Literature Review Examining Broadening Participation in Upper Elementary CS Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 570–576. https://doi.org/10.1145/3545945.3569873

[32] J.E. Sims-Knight and R.L. Upchurch. 1998. The acquisition of expertise in software engineering education. In *FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)*, Vol. 3. 1302–1307 vol.3. https://doi.org/10.1109/FIE.1998.738679

[33] L. Thomas, P. Waterson, and S. Trapp. 2006. Eight Years of Delivering Professional Education and Training for Software Engineering at Fraunhofer IESE: An Experience Report. In *19th Conference on Software Engineering Education and Training (CSEET'06)*. 131–140. https://doi.org/10.1109/CSEET.2006.18

[34] Neena Thota, Gerald Estadieu, Antonio Ferrao, and Wong Kai Meng. 2015. Engaging School Students with Tangible Devices: Pilot Project with .NET Gadgeteer. In *2015 International Conference on Learning and Teaching in Computing and Engineering*. 112–119. https://doi.org/10.1109/LaTiCE.2015.26

[35] R.L. Upchurch and J.E. Sims-Knight. 1997. Integrating software process in computer science curriculum. In *Proceedings Frontiers in Education 1997 27th Annual Conference. Teaching and Learning in an Era of Change*, Vol. 2. 867–871 vol.2. https://doi.org/10.1109/FIE.1997.635990

[36] R.L. Upchurch and J.E. Sims-Knight. 1998. In support of student process improvement. In *Proceedings 11th Conference on Software Engineering Education*. 114–123. https://doi.org/10.1109/CSEE.1998.658307

[37] Arto Vihavainen and Matti Luukkainen. 2013. Results from a Three-Year Transition to the Extreme Apprenticeship Method. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. 336–340. https://doi.org/10.1109/ICALT.2013.104

[38] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. 2011. Extreme Apprenticeship Method in Teaching Programming for Beginners. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 93–98. https://doi.org/10.1145/1953163.1953196

[39] Arto Vihavainen, Thomas Vikberg, and other. 2013. Massive Increase in Eager TAs: Experiences from Extreme Apprenticeship-Based CS1. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) *(ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 123–128. https://doi.org/10.1145/2462476.2462508