Live Coding: A Review of the Literature

Ana Selvaraj
UC San Diego
aselvara@ucsd.edu

Eda Zhang
UW - Madison
rzhang346@wisc.edu

Leo Porter
UC San Diego
leporter@eng.ucsd.edu

Adalbert Gerald Soosai Raj
UC San Diego
gerald@eng.ucsd.edu

ABSTRACT

One of the goals of computing education research is to document the potential strengths and weaknesses of contemporary teaching methods in computing. Live coding has recently gained attention as one of the best practices for teaching programming. To offer a more comprehensive understanding of the existing body of research about live coding, we reviewed papers in computing education research that investigated the value of live coding in an educational setting. We categorized each paper based on (1) how it defines live coding, (2) whether its version of live coding could be considered active learning, (3) the type of study conducted, (4) types of data collected and the data analysis methods used, (5) evidence provided for the effectiveness of live coding, (6) reported benefits and drawbacks of live coding, and (7) reported theoretical frameworks used to explain the basis, effects or goals of live coding. We found that although live coding has been recommended as one of the best practices for teaching programming, there is a lack of empirical evidence to support claims about the effectiveness of live coding on student learning. Finally, we discuss the implications of our findings and suggest future research directions that could develop a more holistic understanding of this pedagogical technique.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

Live Coding; Literature Review; Active Learning; Computer Science Education

ACM Reference Format:
Ana Selvaraj, Eda Zhang, Leo Porter, and Adalbert Gerald Soosai Raj. 2021. Live Coding: A Review of the Literature. In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2021), June 26-July 1, 2021, Virtual Event, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456382

ITiCSE 2021, June 26-July 1, 2021, Virtual Event, Germany.

ACM ISBN 978-1-4503-8214-4/21/06.
https://doi.org/10.1145/3430665.3456382

1 INTRODUCTION

Live coding [14, 31, 36, 39]—the process of writing code live on a computer in front of students during class—is recommended as one of the best practices for teaching introductory programming [5] (along with some well tested techniques like Peer Instruction [33]). The primary purpose of live coding is to teach programming as a process, rather than using static code examples that show programs as a finished product. One perceived benefit of live coding is that it allows the thought process of the instructor to be visible to the students [36, 39]. In addition, live coding can demonstrate well accepted programming practices like incremental coding, debugging, testing and commenting [1, 15, 35, 39].

Live coding has been the subject of many papers over the past two decades (for example, [3, 14–16, 29, 31, 35, 39]). It has been presented as a possible solution to problematic trends observed in beginner programming students, including "Cut & Paste Oriented Programming" and "Random Programming" [15]. Sometimes, it is considered an active learning technique because of its potential to engage students in the coding process [41, 44] and it has been recommended as a best practice to teach programming [5]. However, in some cases, live coding is used in a way that renders students passive while listening to an instructor explain their thought process while coding [3, 45]. Also, the few studies that have tried to determine the effectiveness of live coding for improving student learning have had mixed results [35, 39].

Given these contrasting views on a commonly used instructional technique, there is a need to better understand the range of activities that are considered live coding, its commonly perceived benefits, and to what extent its efficacy has been studied. Apart from understanding the instructor-centric vs. student-centric nature of live coding, it is also important to document the educational theories that are used to explain the basis and/or effects of live coding since they may inform how live coding could best be utilized in classrooms. In this study, we review the literature on live coding to shed more light on this instructional technique and its role in teaching introductory programming.

This paper contributes to the field of computing education by providing a summary of the current state of research on live coding to help guide both practitioners and researchers new to the field. In addition, we offer suggestions about which aspects of live coding should be studied further.

2 RELATED WORK

For the purposes of this paper, live coding refers to any lecture technique where an instructor or student programs in a class with

their code being visible to all students [35]. However, in different contexts, the words "live coding" are overloaded to have the same meaning as the term "live programming" in Software Engineering which is the process of modifying a constantly running program [48]. Furthermore, live coding in music refers to the real-time manipulation of code to create music [2, 40]. These other definitions of live coding are used in the literature study by Rein et al. who compiled the similarities and differences in the values and contributions of the research communities of exploratory programming, live programming and musical live coding [37].

In terms of literature reviews in computing education, Robins et al. reviewed literature related to the psychological/educational study of programming [38]. They stated that live coding could potentially improve a novice programmer's strategic skills. Similarly, there have been literature reviews on student misconceptions in introductory programming [34], the contemporary view of introductory programming drawing from papers published in the years 2010-2016 [25] and pair programming [18].

To the best of our knowledge, there has not been a literature review on live coding in the computing education research community.

3 RESEARCH QUESTIONS

The following are our research questions for our literature review.

(1) *What is live coding?*
(2) *Is live coding an active learning strategy?*
(3) *What are the types of studies (experimental vs. experience reports) that have been published in this area of live coding?*
(4) *What aspects of live coding have been studied by analyzing the types of data collected, and the analysis methods being used?*
(5) *What is the evidence reported for the effectiveness of live coding on student learning?*
(6) *What are the reported benefits and drawbacks of live coding as a pedagogical approach according to the existing body of research?*
(7) *What are the theories reported in live coding research?*

4 METHODOLOGY

For this literature review, we limited our set of considered papers to computing education conference and journal papers in English that study live coding in an educational setting. Two researchers manually reviewed each paper to determine its inclusion or exclusion into the final set of considered papers to reduce risk of human error.

4.1 Paper Search

For our starter set of papers, we used the keywords "live-coding" or "live coding" in the ACM Digital Library on the ACM Full-Text Collection with the applied filters of content-type set to "Research Article" and "Proceeding Series" set to SIGCSE (30 results), ITiCSE (9 results), ICER (3 results), Koli Calling (4 results), SIGITE (3 results) and CompEd (2 results) on January 17th, 2021. Our search in ACM was confined to these six conferences because we wanted our initial list of conference papers to represent the mainstream computing education research community. This search yielded a total of 51 conference papers.

Then, we manually reviewed the 51 papers from the ACM search queries to ensure they were conference papers that discussed or studied live coding as a lecture technique to teach programming in an educational setting. Poster abstracts, short conference papers (1-2 pages), papers on tool design to supplement live coding [7] and papers on the other definitions of live coding related to editing a running program [48] and music [40] were excluded. Moreover, papers that just included the term "live-coding" or "live coding" but did not have live coding as a major component of their study were excluded [28, 30]. This process resulted in a set of seven papers [1, 4, 15, 29, 36, 39, 45] that fit our criteria. This same process was done in IEEE Xplore with the initial search limited to conference papers that used the keywords "live-coding" or "live coding". This yielded 24 results that were all excluded for reasons such as focusing on tool design for live coding [24], being a paper already included in the ACM set with Snowballing [44] (see Section 4.2), being a short conference paper (1-2 pages) [19] or examining live coding done by professional developers rather than instructors and students [23].

4.2 Snowballing

With the initial set of seven ACM papers, we used the snowballing approach to find relevant papers from other conferences. Snowballing is an efficient technique to identify important papers on a specific research area using reference lists and citations [49]. We chose this technique because it helped us discover papers that could be considered the foundation of live coding research [15, 31] (as they are frequently referenced by other papers) and newly published papers from other computing education conferences [35]. For each of the seven papers from our previous step, we reviewed their references and the papers that cited them with help of the "Cited by" function of Google Scholar. Poster abstracts, short conference papers, dissertations, papers that focused on live programming in software engineering, tool design or musical live coding, etc. were excluded. Furthermore, for this snowballing step, we decided to include papers that described a lecture technique that involved writing code in front of a classroom. For example, the author of an influential paper [31] describes live programming as the process of an instructor designing and implementing a large coding project in a classroom. Similarly, the authors of a later study [22] propose a new type of lecture where instructors discuss a software engineering problem and modify source code in front of students while explaining their evaluation of the problem and its potential solutions. These two papers were included because they both describe variants of live coding and may be considered important papers in the live coding research community as they are frequently cited. On the flip side, we decided to exclude an older study [13] because it breaks down programming problems into steps to teach 'programming as a process' that is frequently incorporated into live coding sessions [3, 36] but does not specify a lecture technique where an instructor using this strategy would program in front of students.

Ultimately, this step caused us to discover 13 conference and journal papers on live coding as an addition to the original set of 7 papers from the previous ACM DL search.

5 RESULTS

5.1 RQ 1: What is live coding?

First, it is important to note what the research community considers the definition of live coding. There is disagreement on the minimal requirements of a live coding session when comparing the different definitions of live coding in each paper so we found it prudent to record how live coding is often explicitly defined in research. We evaluate what a paper considers live coding if it explicitly describes its baseline definition of live coding. A common base definition of live coding states that it is the process of designing and implementing a coding project in front of a class during lecture [17, 31, 39, 42]. The term 'coding project' may indicate that the person coding a one-line program is not performing live coding since a small coding exercise is generally not considered a 'coding project'. Two papers [36, 44] mention that live coding must happen from a blank file or 'from scratch' with no skeleton code. This implies that if an instructor had used pre-written skeleton code when coding live in front of a class, they are not 'live coding'. Three papers [20, 36, 44] state that live coding happens when someone is coding on a computer whose screen is projected to the entire class. One paper [35] states the person programming live must think aloud during live coding. Apart from these definitions, another paper [41] uniquely defines live coding as the process where students find a solution to a programming problem through group discussion. These different definitions suggest that there is no strong consensus for what constitutes live coding in education.

Secondly, most studies perform live coding in different contexts, such as recording live coding sessions for students to watch them asynchronously at their convenience [45], to create their own variant of live coding. To further understand the different variants of live coding described in each paper, we highlight three distinct features that captured the similarities and differences across each way live coding is performed below.

5.1.1 How is the source code written? As previously mentioned, whether live coding sessions have instructors and students start from a blank file is not a universal requirement in live coding sessions. A majority of the papers describe whether their variant of live coding had coders write from scratch or not. Table 1 shows a summary of how source code was prepared. The papers, that did not state how their source code was prepared, were omitted from the table.

Table 1: Summary of the different ways that source code is written in live coding sessions

How source code is written	# of studies	Studies that used this method
Only write from scratch	10	[3, 4, 14, 15, 35, 39, 41, 44, 45, 47]
Only write using prepared code	1	[29]
Both from scratch and prepared code	3	[6, 16, 17]

5.1.2 The role of instructors and students. In each paper, we noted who actually coded live in the classroom. Based on that, we classified the studies into 3 possibly overlapping types: 1) Instructor-led, 2) Student-led, and 3) Instructor & Student Collaboration. Most studies regard live coding as an activity that was led by the expert in class (i.e., instructor) where students may passively listen during the entire process without giving their own input and/or asking clarification questions intermittently. A few studies explicitly perform live coding as a student-led activity, where students coded and explained their thought process with their code being projected to the whole class. There are also studies that consider live coding as a collaborative work where teachers take ideas for code from students, and/or students code along with the instructors [39, 44]. The results on who lead the live coding process are summarized in Table 2.

Table 2: Summary of who led the live coding process

Type of live coding	# of studies	Studies that used this method
Instructor-led	12	[1, 6, 15–17, 22, 29, 31, 35, 36, 45, 47]
Student-led	4	[4, 14, 15, 41]
Instructor & student collaboration	4	[21, 39, 42, 44]

5.1.3 Synchronous/Asynchronous. Three papers describe studies where live coding sessions were recorded and presented for students to watch asynchronously [3, 45, 47]. This is important since it means that live coding demonstrations do not necessarily need to be 'live'. These studies studied the value of asynchronous live coding recordings where students had to passively listen. On the other hand, all the other papers describe live coding sessions that happened in physical classrooms where students could ask clarifications questions almost immediately at the very minimum.

5.2 RQ 2: Is live coding an active learning strategy?

Given the varying definitions of live coding and that only some of these definitions meet the standards of active learning, "live coding" may or may not be active learning depending on which variant is being practiced. Active learning is a student-centered paradigm in which students should not only passively listen, but also read, write, discuss, engage in solving problems and actively think and reflect throughout this process. For this review, we will consider a variant of live coding to be active learning if it "consists of short course-related individual or small-group activities that all students in a class are called upon to do, alternating with instructor-led intervals in which student responses are processed and new information is presented" [12]. By this definition, only five of the reviewed papers used a variant of live coding that could be considered active learning [14, 15, 22, 42, 44]. Some of them describe short in-class coding exercises that were given to students after live coding demonstrations by instructors so that the students could apply the concepts they just learned and receive immediate feedback from their peers, instructor or TAs [14, 22, 42, 44].

However, this could also not be considered active learning if the in-class exercises are seen as something separate from the live coding demonstrations or depending on whether feedback is actively given to students rather than just students who explicitly ask for guidance. One paper describes instructor-led live coding followed by a session of student-led live coding where one student has to code in front of the entire class and the rest of the class will provide feedback if necessary. This may not constitute active learning since even though the other students may code along, they do not get feedback [15].

5.3 RQ 3: What types of live coding studies are reported?

The two types of papers we found are: 1) experimental/quasi-experimental, and 2) experience reports. If the paper used within-subjects, between-subjects design or some form of statistical analysis to compare the effectiveness of live coding by comparing it with a control group using another teaching method (e.g., static code examples), then we considered them as experimental/quasi-experimental.

There were only 4 experimental/quasi-experimental studies [29, 35, 39, 47] that studied the effectiveness of live coding using a control group while all other studies reported the instructors' experiences and/or shared students' preferences of live coding using surveys, interviews or open-ended feedback.

5.4 RQ 4: What aspects of live coding have been studied by analyzing the types of data collected, and the data analysis methods being used?

Table 3: Aspects of live coding being investigated by data types.

Aspects investigated	Types of data collected	# of studies	Studies
Student perceptions	Student preferences survey	10	[1, 3, 15, 16, 31, 39, 41, 42, 44, 45]
	Open-ended student feedback	3	[4, 36, 42]
	Interview	2	[17, 47]
Student learning	Grades	5	[6, 16, 39, 41, 47]
	Pre-test & post-test	3	[29, 35, 47]
Cognitive load on students	Cognitive load survey	2	[29, 35]
Level of self-direction of students	The Learning Experience Scale (PRO-SDLS) survey	1	[4]

5.4.1 Types of Data Collected. We found the following types of data were collected in live coding studies: 1) Student feedback surveys (e.g., course evaluation surveys), 2) Open-ended feedback, 3) Pre-test and post-test data, 4) Course grades (exams, quizzes, assignments, projects, overall course grades, etc.), 5) Cognitive Load Survey, and 6) The Learning Experience Scale (PRO-SDLS) survey. Table 3 shows the frequency distribution of the different types of data collected in live coding studies. Note that a single study may collect more than one type of data and therefore the sum of the number of studies column may be greater than the number of papers analyzed on live coding.

5.4.2 Aspects of live coding studied. With a further examination of the types of data collected, we found that the aspects of live coding that had been investigated so far mostly fall into these four categories: student learning, student perceptions, cognitive load on students [35] and self direction of students [4]. Student perceptions refer to the studies that conducted indirect measurements such as students' feedback, students' preferences, as well as self-reported perceived learning. 15 out of the 20 studies investigated student perceptions which is the most common aspect examined. On the other hand, by student learning, we mean studies that used direct measurements such as final grades on students performance and pre-tests/post-tests. This was the second most frequent aspect studied with 8 out of the 20 studies collecting data on this.

5.4.3 Data Analysis Methods Used. The following types of data analysis methods were used to analyze data in live coding studies: 1) Grounded Theory, 2) Statistical tests (e.g., t-test), and 3) Survey analysis. No data analysis was performed in a few papers since no data was collected as part of these studies [14, 21, 22]. In most studies that used the course grades and/or student course evaluation data, no form of inferential statistics was performed. Instead the descriptive statistics of the course grade data or the survey data were presented.

Inferential statistical tests like independent samples t-test and Mann-Whitney U-test were employed in four studies [29, 35, 39, 41] but the specific test used in one of these studies is not reported [29]. Table 4 summarizes the different data analysis methods used.

Table 4: The data analysis methods used in live coding studies are reported in this table.

Data analysis method	# of studies	Studies that used this data analysis method
Survey analysis	10	[1, 3, 4, 15, 16, 31, 39, 42, 44, 45]
Independent samples t-test	4	[35, 39, 41, 47]
No data analysis	3	[14, 21, 22]
Grounded Theory	1	[36]
Mann-Whitney U-test	1	[35]
Data analysis method or statistical test not specified	2	[17, 29]

5.5 RQ 5: Evidence for the (in)effectiveness of live coding

Among the different studies that have been conducted on live coding, only three studies until now have evaluated the effectiveness of live coding on student learning using a control group [35, 39, 47]. Two among them [35, 39] compared live coding with static code examples (a code presentation technique where pre-written code snippets are explained in class). In both these studies, the control group(s) were taught using static code examples while the experimental group(s) were taught using live coding. The same instructor taught both the experimental and the control groups in these studies.

One of these studies [39] was conducted in a regular course where the students' grades in the course on projects, exams, and assignments were used as a measure of student learning. This study reports that students in the live coding group performed statistically significantly better than the students in the static code examples group on the final project. There were no differences between the two groups on other course components.

A more recent study [35] conducted a pre-test and a post-test and used the students' performance on these tests as a measure of student learning. In this study, the static code examples group performed better than the live coding group but the difference was not statistically significant.

Meanwhile, a study [47] compared the pre-test and post-test of student performance scores who were taught using live coding. The authors found that students' improvement in procedural knowledge was significantly better than their gain in conceptual knowledge. However, this study did not compare live coding with static code examples so it does not provide evidence that live coding is more effective than traditional code presentations.

5.6 RQ 6: Benefits and drawbacks of live coding

Live coding has often been perceived to expose programming as a process to students and improve debugging skills [15, 36]. In contrast, it has been reported to have drawbacks including being time-consuming as a lecture technique [35].

To determine the commonly perceived benefits and drawbacks of live coding as a pedagogical strategy, we decided to tabulate the most commonly reported benefits and drawbacks of live coding from each paper over traditional lecture techniques that use static code.

It is also interesting to note that most of these benefits and drawbacks are conjectures drawn from instructors' experiences and student feedback surveys, and are not supported by strong empirical evidence. The strongest evidence for a benefit comes from a study [39] where the higher final project scores of the experimental group, who was taught using live coding over the control group taught using static code examples, could indicate that live coding can ingrain better coding practices in students. However, as stated in the previous section, the difference in test scores between the two groups was not statistically significant in this study and in another experimental study [35]. The perceived benefits and drawbacks of live coding are summarized in Tables 5 and 6 respectively.

Table 5: The six most frequently perceived benefits of live coding across all reviewed studies.

Perceived benefits of live coding	# of studies	Studies which perceived these benefits
Improves debugging skills	9	[3, 4, 15, 31, 35, 36, 39, 41, 44]
Exposes programming as a process	8	[3, 17, 22, 35, 36, 41, 44, 45]
Increases student engagement	7	[6, 17, 31, 35, 36, 39, 42]
Teaches how to apply programming concepts	5	[16, 21, 31, 36, 47]
Improves testing skills	5	[1, 3, 22, 36, 39]
Teaches incremental Coding	5	[3, 22, 35, 39, 44]

Table 6: The perceived drawbacks of live coding across all reviewed studies

Perceived drawbacks of live coding	# of studies	Studies which perceived these drawbacks
Relatively time-consuming	4	[6, 17, 36, 39]
Hard for students to take notes	1	[45]
Hard for students to keep up with the pace of programming	1	[17]

5.7 RQ 7: Theoretical frameworks of live coding

Another feature of the live coding research community is that there are different theoretical frameworks that explain the effects of live coding, inform the goal of live coding as a pedagogical strategy or form the theoretical basis of live coding. These are important to record because it shows the current theoretical foundations of live coding research. As such, we tabulated each theoretical framework involved with live coding and the studies that reference them in Table 7. The depth of each reference is omitted for succinctness as some papers used a reported theory to inform their study design (for example, to create and test a new variant of live coding [22]) while others only mention the theory briefly to give minimal context [39]. Moreover, there are some papers that did not reference any theoretical framework at all [1, 45].

Although Cognitive Apprenticeship (CA) is the most common theoretical framework referenced, most studies referencing CA only study live coding as part of the modeling phase while ignoring the other phases in CA (scaffolding and fading). There were two exceptions to this trend [4, 22].

Furthermore, there are many theories that are only mentioned in passing but its relation with live coding are never studied extensively like Discovery Learning [14] and Just-In-Time Learning [44].

Table 7: The theoretical frameworks used in live coding across all reviewed studies

Theoretical frameworks of live coding	# of studies	Studies that reference this theoretical framework
Cognitive Apprenticeship	5	[4, 22, 35, 36, 39]
Cognitive Load	2	[29, 35]
Constructivist Educational Approach	2	[4, 14]
Constructive Alignment Theory	2	[14, 15]
Apprenticeship	1	[15]
Discovery Learning	1	[14]
Studio Teaching	1	[6]
Modality Theory	1	[29]
Just-In-Time Learning	1	[44]

6 DISCUSSION

Effectiveness of live coding on student learning. One of the key observations we made from our literature review is that, although live coding has been discussed and used as one of the common code presentation techniques in computing education, most of the published studies on live coding have focused primarily on students' perceptions and preferences of this teaching method. Very few studies [35, 39] have attempted to evaluate the effectiveness of this teaching method on student learning by comparing it with static code examples (i.e., presenting pre-written code examples in-class). Prior studies in physics education research have shown that in-class physics demos did not generally contribute to student learning on their own unless students were asked to predict the output of those demos [10, 26, 27]. Given the paucity of empirical studies on live coding and the mixed results from the existing few empirical studies, further research on the topic would be beneficial to the community before recommending the practice.

Rethinking active learning in CS. Many CS education researchers and practitioners believe that instructor-led live coding is an active learning strategy because students make continuous predictions about the code they are witnessing in a live coding session [6, 35, 41, 44], however similar arguments might be made about lecture in general. In contrast, others report that only student-led versions of live coding can be considered active learning since students are only actively involved when they are writing code themselves [14, 15]. One challenge for answering this question is the lack of clear definition for active learning [11]. We recommend future research seek to understand what components of live coding are necessary for it to actively engage students (e.g., through student observations, interviews, etc.). Secondly, if live coding is viewed to be an active learning pedagogy, studies aiming to understand its efficacy should compare live coding against other effective active learning strategies in CS such as Peer Instruction [32, 33, 43].

The neglected phases of Cognitive Apprenticeship in live coding. On a related note, we also found that even though most studies on live coding refer to Cognitive Apprenticeship (CA) [8, 9] as the theory behind live coding, most of the studies using instructor-led live coding [31, 35, 39] primarily focuses only on the first phase of CA, i.e., modeling. Only a few studies on live coding use the second phase of CA (i.e., scaffolding) using techniques like in-class coding where students write code in the presence of instructors and TAs [42, 44] and student-led live coding [14, 15]. The third phase of CA (i.e., fading) is usually achieved by means of programming assignments which are intended to make students write code independently. We recommend future live coding studies to focus more on the scaffolding phase as we believe it could create a safe space for students to make mistakes while writing code in the presence of an expert.

Studying instructors' perceptions on live coding. Although instructor-led live coding is the most commonly reported form of live coding in the literature, it is interesting that there were no studies that captured the instructors' perception on this technique. In instructor-led live coding, the instructor is more actively involved in the process (by typing, compiling, debugging, testing, thinking-aloud, etc.) than the student so it may be interesting to study the cognitive load [46] imposed by live coding on instructors.

Limitations. While searching for papers to review, we decided to exclude papers on the design of tools to supplement live coding [7]. This is because we wanted to examine papers that primarily focus on studying live coding in an educational setting. If we decided to include the papers on tool design, our review may have been more comprehensive in capturing the entire body of live coding research. Another limitation is we only examine 20 papers compared to other literature reviews which cover many more papers. This is because live coding is much less extensively studied than other common literature review topics like introductory programming [25] and student misconceptions in introductory programming [34]. However, we still believe that our review could be useful since it summarizes previous live coding studies and notes unexplored perspectives that could improve the utility of live coding as an educational tool.

7 CONCLUSION

In this paper, we reviewed conferences and journal papers that studied live coding in classrooms. We found that there is no consensus on a universal definition of live coding. We also discovered that most research papers on live coding were experience reports that used feedback from student surveys to infer the effects of live coding. The few empirical studies that do exist have limitations and do not provide strong evidence for the commonly reported benefits of live coding. This finding should be a call for more empirical studies exploring the benefits of live coding over traditional lectures. Furthermore, because most claims about live coding are based on student feedback, an unexplored area of research is how live coding impacts instructors and how they perceive the practice. In sum, this article provides a holistic summary of the state of live coding research by examining the conference and journal articles that study live coding in computing education.

ACKNOWLEDGMENTS

We greatly appreciate the reviewers for their helpful feedback. This work was supported in part by NSF award 2044473.

REFERENCES

[1] Maurício Aniche, Felienne Hermans, and Arie van Deursen. Pragmatic software testing education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 414–420, 2019.

[2] Renick Bell. A live coding improvisation. In *Proceedings of the 9th ACM Conference on Creativity & Cognition*, pages 392–393, 2013.

[3] Jens Bennedsen and Michael E Caspersen. Revealing the programming process. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 186–190, 2005.

[4] Naomi R Boyer, Sara Langevin, and Alessio Gaspar. Self direction & constructivism in programming education. In *Proceedings of the 9th ACM SIGITE conference on Information technology education*, pages 89–94, 2008.

[5] Neil CC Brown and Greg Wilson. Ten quick tips for teaching programming. *PLoS computational biology*, 14(4):e1006023, 2018.

[6] Russel E Bruhn and Philip J Burton. An approach to teaching java using computers. *ACM SIGCSE Bulletin*, 35(4):94–99, 2003.

[7] Charles H Chen and Philip J Guo. Improv: Teaching programming at scale via live coding. In *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale*, pages 1–10, 2019.

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

[9] Allan Collins, John Seely Brown, and Susan E Newman. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *Thinking: The Journal of Philosophy for Children*, 8(1):2–10, 1988.

[10] Catherine Crouch, Adam P Fagen, J Paul Callan, and Eric Mazur. Classroom demonstrations: Learning tools or entertainment? *American journal of physics*, 72(6):835–838, 2004.

[11] Valerie Drew and Lorele Mackie. Extending the constructs of active learning: implications for teachers' pedagogy and practice. *Curriculum Journal*, 22(4):451–467, 2011.

[12] Richard M Felder and Rebecca Brent. Active learning: An introduction. *ASQ higher education brief*, 2(4):1–5, 2009.

[13] Rex E Gantenbein. Programming as process: a "novel" approach to teaching programming. *ACM SIGCSE Bulletin*, 21(1):22–26, 1989.

[14] Alessio Gaspar and Sarah Langevin. Active learning in introductory programming courses through student-led "live coding" and test-driven pair programming. In *International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL*, 2007.

[15] Alessio Gaspar and Sarah Langevin. Restoring" coding with intention" in introductory programming courses. In *Proceedings of the 8th ACM SIGITE conference on Information technology education*, pages 91–98, 2007.

[16] Nasser Giacaman. Teaching by example: using analogies and live coding demonstrations to teach parallel computing concepts to undergraduate students. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 1295–1298. IEEE, 2012.

[17] Tor-Morten Grønli and Siri Fagernes. The live programming lecturing technique: A study of the student experience in introductory and advanced programming courses. In *Norsk IKT-konferanse for forskning og utdanning*, number 4, 2020.

[18] Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. Pair programming in education: a literature review. *Computer Science Education*, 21(2):135–173, 2011.

[19] Hui-Chun Hung. Flipped learning with live-coding approach for programming concepts learning. In *2018 1st International Cognitive Cities Conference (IC3)*, pages 223–224. IEEE, 2018.

[20] Roger T Johnson and David W Johnson. Active learning: Cooperation in the classroom. *The annual report of educational psychology in Japan*, 47:29–30, 2008.

[21] Luke Johnston, Madeleine Bonsma-Fisher, Joel Ostblom, Ahmed Hasan, James Santangelo, Lindsay Coome, Lina Tran, Elliott de Andrade, and Sara Mahallati. A graduate student-led participatory live-coding quantitative methods course in r: Experiences on initiating, developing, and teaching. *Journal of Open Source Education*, 2(16):49, 2019.

[22] Michael Kölling and David J Barnes. Enhancing apprentice-based learning of java. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 286–290, 2004.

[23] Jan-Peter Kramer, Joachim Kurz, Thorsten Karrer, and Jan Borchers. How live coding affects developers' coding behavior. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 5–8. IEEE, 2014.

[24] Francesco Maiorana, Daniela Giordano, and Ralph Morelli. Quizly: A live coding assessment platform for app inventor. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, pages 25–30. IEEE, 2015.

[25] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90, 2018.

[26] Kelly Miller, Nathaniel Lasry, Kelvin Chu, and Eric Mazur. Role of physics lecture demonstrations in conceptual learning. *Physical review special topics-physics education research*, 9(2):020113, 2013.

[27] Marina Milner-Bolotin, Andrzej Kotlicki, and Georg Rieger. Can students learn from lecture demonstrations. *Journal of College Science Teaching*, 36(4):45–49, 2007.

[28] Mia Minnes, Christine Alvarado, Max Geislinger, and Joyce Fang. Podcast highlights: Targeted educational videos from repurposed lecture-capture footage. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 365–371, 2019.

[29] Briana B Morrison. Dual modality code explanations for novices: Unexpected results. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, pages 226–235, 2017.

[30] Ayesha Naeem Syeda, Rutwa Engineer, and Bogdan Simion. Analyzing the effects of active learning classrooms in cs2. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 93–99, 2020.

[31] John Paxton. Live programming as a lecture technique. *Journal of Computing Sciences in Colleges*, 18(2):51–56, 2002.

[32] L. Porter, C. Bailey-Lee, B. Simon, Q. Cutts, and D Zingaro. Experience report: A multi-classroom report on the value of peer instruction. *ITiCSE*, 2011.

[33] Leo Porter, Dennis Bouvier, Quintin Cutts, Scott Grissom, Cynthia Lee, Robert McCartney, Daniel Zingaro, and Beth Simon. A multi-institutional study of peer instruction in introductory computing. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 358–363, 2016.

[34] Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–24, 2017.

[35] Adalbert Gerald Soosai Raj, Pan Gu, Eda Zhang, Jim Williams, Richard Halverson, and Jignesh M Patel. Live-coding vs static code examples: Which is better with respect to student learning and cognitive load? In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 152–159, 2020.

[36] Adalbert Gerald Soosai Raj, Jignesh M Patel, Richard Halverson, and Erica Rosenfeld Halverson. Role of live-coding in learning introductory programming. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, pages 1–8, 2018.

[37] Patrick Rein, Stefan Ramson, Jens Lincke, Robert Hirschfeld, and Tobias Pape. Exploratory and live, programming and coding: A literature study comparing perspectives on liveness. *arXiv preprint arXiv:1807.08578*, 2018.

[38] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.

[39] Marc J Rubin. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 651–656, 2013.

[40] Alex Ruthmann, Jesse M Heines, Gena R Greher, Paul Laidler, and Charles Saulters. Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 351–355, 2010.

[41] Amy Shannon and Valerie Summet. Live coding in introductory computer science courses. *Journal of Computing Sciences in Colleges*, 31(2):158–164, 2015.

[42] Madhav Sharma, Surya Ayyalasomayajula, Nikunj Dalal, et al. Teaching programming to the post-millennial generation: Pedagogic considerations for an is course. *Journal of Information Systems Education*, 31(2):96–105, 2020.

[43] Beth Simon, Julian Parris, and Jaime Spacco. How we teach impacts learning: peer instruction vs. lecture in CS0. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 2013.

[44] Adalbert Gerald Soosai Raj, Jignesh Patel, and Richard Halverson. Is more active always better for teaching introductory programming? In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 103–109, 2018.

[45] Ben Stephenson. Coding demonstration videos for cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 105–111, 2019.

[46] John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312, 1994.

[47] Sheng-Rong Tan, Yu-Tzu Lin, and Jia-Sin Liou. Teaching by demonstration: programming instruction by using live-coding videos. In *EdMedia+ Innovate Learning*, pages 1294–1298. Association for the Advancement of Computing in Education (AACE), 2016.

[48] Steven L Tanimoto. A perspective on the evolution of live programming. In *2013 1st International Workshop on Live Programming (LIVE)*, pages 31–34. IEEE, 2013.

[49] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10, 2014.