More Effective Contextualization of CS Education Research: A Pair-Programming Example

Briana Bettin
Michigan Technological University
Houghton, MI, USA
bcbettin@mtu.edu

Linda Ott
Michigan Technological University
Houghton, MI, USA
linda@mtu.edu

Leo Ureel
Michigan Technological University
Houghton, MI, USA
ureel@mtu.edu

ABSTRACT

This position paper discusses the need for greater inclusion of context in papers describing computer science education research. This inspiration arose from our efforts to compare our experiences with pair programming in an introductory computer science course with experiences described in the literature. We quickly observed that the behaviors associated with the term "pair programming" and the contexts can differ greatly between universities and yet the phrase pair programming is often used with no further explanation.

A brief literature survey is used to demonstrate differences in the implementation of pair programming and the context that might impact the results. We identify attributes that are likely appropriate for much CS education research, as well as specifically consider relevant attributes for research involving pair programming. This anchors our paper and demonstrates specific attributes that require consideration beyond the general computer science classroom.

Our goal is to foster conversations on providing appropriate context in computer science education research. We argue that by providing such context, studies can be more easily replicated or distinguished, a greater understanding of attributes influencing the research can be gained, and other educators can more easily determine the relevance of the research to their classroom environment.

CCS CONCEPTS

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

KEYWORDS

Computer science education research, CS1, pair programming, student perception, context, attributes

ACM Reference Format:
Briana Bettin, Linda Ott, and Leo Ureel. 2019. More Effective Contextualization of CS Education Research: A Pair-Programming Example. In *Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland Uk.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3304221.3319790

ITiCSE '19, July 15–17, 2019, Aberdeen, Scotland Uk

ACM ISBN 978-1-4503-6301-3/19/07…$15.00
https://doi.org/10.1145/3304221.3319790

1 INTRODUCTION

As computer science educators, we have often seen a line similar to "our CS1 course uses pair programming" in papers we have read. This can feel sufficient at first, but this is because of our own interpretations of what that means. We may visualize our own classrooms that use pair programming. We may assume an ideal lab for pair programming, such as those described in well-known papers. Or perhaps we expect something else entirely - perhaps based on a lab seen in visits to industry.

The words "pair programming" appear to convey a complete concept, but there is much context missing. By providing more implementation detail, we can better reflect on the experiences we create for our students. Our readers can also better conceptualize the environment surrounding our work. This can help us to find research that is similar to our context, to recognize differences in implementations when we work to replicate research, or to understand what may appear on the surface to be contradictory findings.

There is need for context in any academic writing, and examining what is meant by "our classroom uses pair programming" is a start in the computing science education space. There are many aspects to what we mean by context. Some may be more critical than others for specific studies, and far too many exist to include them all. Our position here explores initial ideas for what may be useful in computing education papers. Specifically, we explore what additional information would aid in understanding the pair programming context, much of which overlaps with the larger context of computing education. Our hope is that this motivates discussion on what context is needed in computing education research.

2 MOTIVATIONS

When describing our own courses, we have used the phrase "our course uses pair programming". As we've explored the literature on pair programming, we realized the need for greater clarity in what we and others mean by that phrase. For example, the computer lab we use is not optimized for pair programming. When our students do not follow the pair programming practices we've taught, a source of difficulty may be our lab set-up. Our lab has 40 computers each with an individual keyboard and mouse. The computers are on tables grouped in pairs. Each student could have a computer and often students become frustrated when told to share one computer shoulder-to-shoulder in classic pair programming style. Many of their complaints are valid: "It's easier to have the problem set open on this screen so we can see our objectives quickly!" and "If we switch roles but simply swap the working computer, we aren't tripping over each others' backpacks!". Allowing these reasonable behaviors often results in students sliding into habits that are inconsistent with sharing context on a single computer.

In discussions with colleagues elsewhere [2], it became apparent that the classic definition of pair programming is not always what individuals mean when they say "pair programming". Indeed the discussions made clear that student behaviors we were considering as divergent, others believed to fall under the definition of pair programming. Given that pair programming behaviors are not standard across our classrooms - but as a community, we often write as though they are - we felt compelled to begin this discussion.

A lack of sufficient description of context in computing education research is not restricted solely to research on pair programming. A 2018 ITiCSE working group [9] conducted a major systemic literature review on introductory programming in tertiary institutions. One finding was that many of the 735 papers analyzed did not provide adequate context for other educators to assess the relevance of the research findings to their institution and/or to replicate the research study. Suggested context included demographic details and syllabus. We attempt to further this discussion here by recommending details to be considered in adding context. The focus on pair programming research is meant to demonstrate that additional context may be needed in specific research areas.

3 BACKGROUND

In order to identify potential attributes that may be valuable in creating context, we examined papers describing pair programming studies with background context in mind. The collection of papers is not meant to be comprehensive, but to provide insight into what is reported as contextual information and what may be missing.

Papers selected focused on pair programming, with many using pair programming as the primary tool to answer a research question - such as "does pair programming affect X?". Each paper chosen described the methodology with detail beyond simply stating pair programming occurred. Several were also chosen for their appearance as citations in numerous other pair programming works - highlighting the papers' influence. The papers we discuss presented interesting and often unique contextual information. Also highlighted in our discussion are questions that lingered after reading - such as information that was presented in other papers, or largely appeared to be missing but may be of value.

Work in pair programming has already explored several unique factors - personality types, skill levels, and confidence are just a few. By adding context to work such as this, data trends that may not have been visible previously can surface. It may also yield more new hypotheses: the traditional results may hold true, but perhaps certain contexts benefit more from specific modifications. Indeed, in other disciplines [3, 17, 20, 22] reviewing the similarities and differences in fidelity of implementation for classroom interventions has proven valuable in assessing what core aspects are critical to an implementation, and what may change among contexts. Without first establishing this context in our papers however, we cannot begin to design such reviews, which allow us to better understand and implement our interventions.

3.1 Lab Design

In a reflection of pair programming work by Williams [24], background is provided on their university lab setup, including a diagram of the table layout. The lab is noted to have been designed for pair programming - with each computer having two monitors, two keyboards, and two mice. This allows the reader to visualize the space in which these studies are occurring and immediately recognize differences from their own space. The pair programming space may impact student perception or success of the activity. Williams [24] has noted that students should not need to "get up" to switch roles. In other works by Williams [25, 26] however, this context is not fully specified. If one reviews only specific papers by the author, this valuable lab design information may be missing. When working with collaborators at different universities [25], distinctions between the universities were also not presented.

Chong and Hurlbutt [5] conduct their research in an industry setting, but also provide valuable insight into the setup of the pair programming work-space. Their study is conducted between two separate "teams", and the difference in team spaces, from both layout of desks to setup of machines, is documented.

Beck [1] describes "one computer, one mouse, one keyboard" as ideal. Clearly the lab Beck describes and the lab Williams [24] describes differ, even if this distinction might be considered slight. Both make strong arguments for lab design playing a role. Both argue for pair programming's effectiveness in proximity and shared context of programmers - yet their well-documented designs differ.

Questions about lab design become apparent in further studies [11, 16, 19]. Each of these studies provides great context in other areas, but misses discussion on the actual design of their space. In Thomas et al. [19], Beck's proposed space design is referenced in describing pair programming, but the actual space used is not described. One might assume their lab is designed in a similar manner to Beck's, but this is speculative. Simon and Hanks [16] review pair programming attitudes at two different institutions, but do not describe similarities and difference in lab setup. McDowell et al. [11] describe time spent in the lab space and expectations, but do not detail the layout of the space.

If a lab is not built for pair programming, it is likely that there is a computer for each chair. While students may be encouraged to work at one computer, this setup is distinct from those of Williams and Beck. Instructors are likely to monitor and encourage expected behavior, but the space itself allows for behavioral differences. These differences in lab design may reveal new questions or insights based on the different setups. The layout of the space may impact research results, but we can only determine this if the space is clearly described.

3.2 Pairing Methodology

Several papers center discussions on methods for choosing effective student pairs. Even papers for which this is not a focus often provide some insight into this space.

In some papers, students are paired by personality traits [14, 28]. In others, students are paired based on confidence or skill levels [19, 27]. Williams et al. [24, 26] note several possible approaches that have been studied in their labs. They find that some of the best compatibility options may be midterm scores, pairing together a Myers-Briggs sensor and intuitor, or similar work ethic. Pairings of these kinds can be created through data collection and compatibility assessments, and the use of these tools can greatly aid in pairing consistency for instructors.

McDowell et al., as well as Williams and Kessler [11, 25], describe processes in which students are able to have some choice in their partner selection. This approach appears to align with work suggesting that comfort between group members and self selection of groups can best motivate confidence, responsibility, and success in all members [12, 18].

Pairing methodology appears to be one of the most widely covered topics in the pair programming literature. Much of the pair programming research includes the pairing methodology as an attribute. Even when the pairing methodology is not the focus of the study, the value of it in describing context seems to be widely accepted based on the frequency of its reporting. Novel methods of pairing are likely to impact results, and as there are many potential methods for pairing, it would be difficult for a reader to assume a given scenario without one being explicitly described.

3.3 Demographics

As stated, many papers describe in detail their pairing methodologies. Goals may often refer to a specific demographic, such as the encouragement of women in computing. Details on the student population of the institution and even the classroom itself, however, are often omitted.

The omissions of such details may be due to an assumption that the demographic of the university is not relevant. However, research shows that sense of belonging is highly critical to student retention and perceptions of computing [18, 21]. Diversity efforts in pair programming papers have focused largely on positive impacts for women, such as Werner et al.'s work [23], but other underrepresented groups are often not noted unless they are the study's primary focus.

These details may not appear relevant to non-focused studies, but understanding class composition can impact understanding the role sense of belonging may play in the pair. When working in closely knit groups, students of underrepresented groups may feel an amplified disconnect, which may affect belonging and performance [6]. Observed interventions, such as those by Scott et al. [15] with underrepresented groups, have shown positive correlations with their persistence and performance in computing despite initial marginalization. Focused studies showcase that demographic factors can greatly impact students. Thus, a better understanding - even at a high level - of demographics may increase our understanding of pair programming research.

Even when not being observed in pairing research, personality differences may be valuable demographic information. Layman [8] finds introverted, reflective learners who are confident in their skills are more likely to be reluctant to pair. This is not to say they will not benefit, but that they may not warm up to the idea of pairing easily, which may decrease a partner's sense of belonging. This result was also noted by Thomas et al. [19] when observing pairs, where students are coined as "Code Warriors" who are highly confident, and are noted to have more frustration with pairing. A classroom largely comprised of students exhibiting these traits may result in different observations than one that differs - such as being full of extroverted learners who are not as confident in their skills. This indicates that not only is pairing methodology relevant, but that additional details are valuable in illustrating sense of belonging and willingness to participate.

Primary student year and non-major enrollment may affect the classroom population as well. McDowell et al., as well as Williams and Kessler [10, 25], make note not just of the course but the year students enroll in it. This is valuable - a CS1 course with a significantly high number of third-year non-major students may be distinct from a CS1 course that is almost exclusively first-year major students. For example, students in later years likely have developed stronger study habits and more effective collaborative skills, even if they lack confidence in programming. First-year students, on the other hand, are primarily relying on collaborative skills learned doing projects in K-12 and so may exhibit different behaviors. Non-majors may also have unique motivations for pursuing a computing course, which may affect their perception or performance. Without making clear not just what course we are teaching, but what type of students are enrolled, we cannot parse appropriate observations.

Details of demographics may be valuable not only at a classroom level, but at a departmental and university level. If the course's composition differs from the overall department, this may have relevance to observations on retention or motivation. If the university differs from the department or course, students may feel a heightened or lessened sense of belonging in the computing space.

Location and size of the university or department may also be relevant. A large university in an urban setting and a small rural university have quite different contexts for numerous reasons, and it is likely students choosing to attend each have different motivations, experiences, and perceptions. Location may also relate to other demographic attributes that may be of interest. Some universities may have greater populations of non-native speakers, nontraditional students, or students needing accommodations. Having this information allows educators to aggregate and review in order to determine what might affect the student experience in their classrooms.

The lack of a discussion about demographics may be because we largely expect that our contexts do not differ. However, not stating this information also means we do not challenge this assumption. Clearly all academic environments are not the same. When made concrete we are better able to view a more comprehensive picture of our teaching practices.

3.4 Pairing Behavior

Largely the original catalyst behind this paper was actual pairing behaviors. These are often difficult to discuss or observe, but may be vital for greater understanding. Although we may provide students with a definition of pair programming that we expect them to follow, we know students do not always perform as anticipated.

Pair programming is described by Williams [24] as two programmers working at one computer on the same problem. Two roles are expected to exist: the driver, who implements the design, and the navigator, who plans and oversees the design implementation. It is widely acknowledged that pair programming as a practice is effective in student learning [13]. But to fully understand what really is effective when students are pairing, any changes in their behavior from our assumptions needs to be described.

Chong and Hurlbutt [5] note shifts in expected pairing behaviors quite clearly. In particular, they highlight that while "driver" and "navigator" roles are widely acknowledged, groups often do not perform these roles as anticipated. Bryant's [4] role analysis shows the behaviors are traded off somewhat erratically. Chong and Hurlbutt [5] also found that the language is so deeply pervasive, asking qualitative questions will often not be enough to distinguish if role divergence has occurred. The "driver" and "navigator" terminology were utilized despite deviation - respondents believed they had behaved in these roles as expected. Simon and Hanks [16] note that students may describe these roles, but not actually use the driver/navigator terms.

One of pair programming's assumed benefits is context sharing [5, 16]. Both students viewing the same code at the same point have a grounded basis of operation. The benefit is improved debugging and collaborative problem solving to increase code quality and production. Students may sometimes remove themselves from this shared context by altering the assumed roles. For instance, a navigator may take on the role of researcher. In this role, the navigator searches for helpful materials while the driver continues to test different ideas while waiting for the navigator's feedback.

In labs with a computer to each student, this researcher role may result in the two students on separate machines [2]. Students are working the same problem and likely the same aspect of the problem - but the navigator is no longer guiding the driver and assisting in debugging. Instead, they are working to solve the problem in separate contexts. In labs with only a single machine to a pair, this may still occur with the navigator using a separate device such as a cell phone or simply reviewing a textbook.

This behavior is not inherently negative. When students are stuck, reviewing materials or identifying code patterns to replicate and modify can be an extremely useful learning tool. However, this is not the behavior described in the definition we often assume for pair programming, which emphasizes the benefits of a consistent shared context. While students may largely share the context together, divergence can certainly exist. Acknowledging this behavior may yield new research questions, such as what causes students to separate from sharing context and at what point do they return. This may provide greater understanding of what we mean by pair programming as well - perhaps we may discover benefits to this behavior and thus feel comfortable allowing it in our classrooms.

In our own observations [2], we noted not only the researcher role exhibited, but a "leader and follower" role, in which the driver is leading the coding efforts with the navigator building identical code alongside on their own machine. As an anecdote, some students showing this behavior felt they were not "getting enough of a chance to practice writing code" when their time was halved between driver and navigator. This variant of the roles was their solution that still encompassed sharing the same problem context and discussing/debugging as they built, but working on distinct machines. Students also exhibited what could be called "race and regroup", in which both worked on the same problem separately, reconvening to share ideas for the best method of moving past obstacles. Both students would agree upon the optimal solution for submission, but developed separately and collaborated primarily at critical points. This particular behavior was described to some degree by Williams and Kessler [25] as partners doing simple coding and testing separately. We also observed groups that would begin their work as pairs but over time be seen working entirely separately. These groups may even be observed attempting to "divide and conquer", with both students working on entirely separate problems, thus sharing no context.

In our case we frequently intervened to remind students of the "optimal" pairing strategies, but students often fall back to the observed collaborative habits. Perhaps these behaviors are not negative, as Chong and Hurlbutt [5] suggest. There may indeed be multiple forms of effective pairing behaviors, as we often see teams behave differently and yet achieve similar success. Indeed, Williams and Kessler [25] note that students may recognize that certain parts of their process may benefit more from a combined effort. This may be so persistent that for some researchers, certain behaviors may be completely expected during pair programming, without it being recognized that it is a variation. Without acknowledging observed variations in behavior, we cannot begin to understand the positive or negative effects these behaviors may have.

Pairing behavior also encompasses areas such as partner dynamics. Domination in the partnership can result in the non-dominant student suffering both in confidence and performance [19, 26, 27]. This research shows the importance of recognizing divergent behaviors - to distinguish the beneficial from the harmful and find ways to motivate positive practices for all students.

It can be difficult without a focused study to clearly assess pairing behavior. However, simply beginning to discuss these behaviors starts providing context and uncovering further research questions. Perhaps students pair program as expected at the beginning of the course, but toward the end changes in behavior can be seen. Perhaps the pair programming expectations of the course allow for students to exhibit differing behaviors. Stating this information allows readers to not assume the theoretically perfect pair programming behaviors when that may not be the reality.

3.5 Course Structure

Course structure may impact student perceptions, performance and how they work with their partner. Course design concerns of particular interest to pairing include how much pair programming students are assigned, where they are observed pairing, and the portion of their grade influenced by the pair programming activities.

In some courses such as Williams's [24], students primarily do individual work with a subset of their work (such as lab work) being done with pair programming. In other courses such as McDowell et al.'s [11], nearly all work, except for exams, is done using pair programming. There are certainly many more variations beyond these, but these pairing expectations may impact the research results.

Further, the space in which pairing is observed (if at all) is important. A general assumption may often be a supervised lab space working on specific problem sets such as what Williams [24] describes, but this is of course speculative if not stated. If pairing behavior occurs in unsupervised contexts, in non-required labs such as McDowell et al. [11], or remotely such as Hanks [7], the results may be affected.

Languages taught, material covered, and tools utilized provide greater understanding of the course and allow for exploration of

potential interactions. Some papers describe the content covered [10, 25, 27]. The degree of detail varies, but all note aspects of course content and expectations. Others describe the type of course, but not the material presented to students [11, 16, 19]. Perhaps a collaboration tool assists students in knowing when to switch roles, or a unique topic is taught. Noting this allows an understanding of uniqueness between seemingly similar contexts, where "a CS1 course" does not. Without it, assumptions are likely made - perhaps such as that what is left unsaid mimics our own context.

How students are taught about pair programming may impact their understanding and implementation of the process. Students may be taught through a short presentation/lecture [25, 27] or by reading a paper [11, 25].

Students partnership may stay the same through a course, or may rotate several times through the semester as Williams [24] describes. Some, such as Wood et al. [27] change as frequently as every session. This can change the formation and expectations of teaming behaviors. There may also be expectations of when students swap roles - such as after set time periods, or after specific problems. These choices all may alter the environment and perceptions surrounding pairing in a course.

4 CONTEXT ATTRIBUTES

Our review of the literature suggested context that is likely broadly relevant to computer science education research along with context specific to pair programming. Below is suggested context attributes for consideration. Many include example attribute values for clarification.

4.1 Context for CS Education Research

University Attributes

(1) The student population consists of students who are:
([specify ethnicity and race]; [specify gender]; Commuter/Residential/Remote; Non-traditional; Non-native language speakers; Extroverted/Introverted; Neurotypical/Neurodiverse; First generation ...)
(2) The university has: ( ) students
(3) The university location is largely: (Urban; Rural)
(4) The faculty population of the university when compared to students: (Largely matches; Varies in terms of...)

Department Attributes

(1) The department's student population when compared to the university: (Largely matches; Varies in terms of...)
(2) The students who persist in computing to graduation when compared to the department: (Largely matches; Varies in terms of...)
(3) The department's faculty population when compared to the university: (Largely matches; Varies in terms of...)

Course Attributes

(1) The student population that enrolls in this course when compared to the department: (Largely matches; Varies in terms of...)
(2) Non-major enrollment in this course is approximately: ()%
(3) Non-major enrollment in this course may be influenced by the fact that it:
(Can fulfill a general education requirement by taking the course if they are non-majors; Is required for a large portion of non-majors in their chosen major)
(4) The course is: (CS0; CS1; CS2; ...)
(5) Most students are (first, second, ...) year students:
(6) The course content consists of:
([specify languages used]; [specify topics covered])
(7) Tools used in the course include:
([specify Programming environment], [specify submission system/classroom management tools], [specify any course tie-ins such as polling software], [specify any collaboration tools]; [specify any visualization tools])
(8) Student work is:
(Entirely individual; Entirely pair work; Entirely in teams; Mostly individual, with pair programming in some projects; Mostly collaborative, with individual assessments)
(9) Research related to this course is being used:
([To change curriculum for a specified purpose], [To observe habits for a specified purpose])
(10) Additional resources available to students for this course include:
(Scheduled time with a tutor; A learning center with walk-in hours; Discussion forums with other students; Retry or mastery based grade systems)

4.2 Context for Specific Research

Pair Programming Attributes

(1) The room in which students pair program is set up such that:
(Each student has their own computer; There are two keyboards controlling one computer; There is one computer and keyboard for a pair of students)
(2) When students pair program, they are expected to work:
(In a supervised lab section on specific problems; In a supervised lab on any assigned work; Collocated, but are unsupervised; Remotely)
(3) Students are taught the following about pair programming:
(It is an industry standard; Communicating code ideas is a marketable job skill; There are expected driver and navigator roles; Only one computer should be used; Roles should switch often)
(4) Students are taught about pair programming practices by:
(Reading a publication; Watching an instructional video; Listening to an explanation or lecture; Observing a skit; Diving in to work together with interventions as needed
(5) Students pairing decisions are made by:
(Allows students to choose their own partner; Matches them with a partner whose personality is [similar/different]; Matches them with a partner whose skill level is [similar/different]; Matches them based on another assessed trait [specify]; Partners are chosen at random by instructor; Partners are based upon where students first sit)

(6) Student partners change: (Never; Each week; Every other week; After major assignments; Halfway through the course; If changes are requested or voted on)
(7) Students are expected to swap driver and navigator roles: (After each problem; After a set time interval; Whenever the pair decides but non-swapping is intervened upon; Whenever the pair decides with no interventions)

4.3 Example Context Statement

Below is an example using a fictional university to illustrate how this seemingly large addition of context can be approached. This information might be included in a paper's background section, or broken apart with relevant information in specific sections.

Our university has 23,000 students and is located in a rural area of the United States. The majority of the student body are white males who live on campus. Approximately 20% of our students are indigenous females who commute and 5% are non-native language speaking students who take courses remotely. The students in our department differ from the campus as a whole, being primarily remote non-native language students. The faculty on our campus are primarily white females, and our department matches this trend as well.

In our CS1 course, students more closely resemble the university as a whole, with non-major enrollment being approximately 60% as many non-majors are required to take CS1. Students are typically in their second year. We cover introductory concepts in Python using PyCharms, and assign work via Blackboard. Our students collaborate using pair programming on all assignments, except for assessments. A tutor is available for students to schedule help sessions. The investigators for this research are not the instructors of the course. The goal of this research is to observe student habits of communication while pair programming.

In our classroom, there is one computer to each pair of students, who work in a supervised one-hour lab on assigned coursework. Pair programming is introduced to students with an instructional video, which discusses the driver and navigator roles and teaches them that they should switch roles often. Students are paired based on similar personality types, which are assessed using OCEAN. Partners change halfway through the course. Students are encouraged to swap roles whenever they complete a problem, but are not forced to.

4.4 Considerations

We listed a broad range of attributes and have attempted to suggest values without judgment. For instance, in describing pair programming behaviors, a range of behaviors are described. Some may be considered positive, while others negative. Noting both without judgment allows for ultimately arriving at a better understanding of which behaviors do actually matter.

In addition to the context attributes we've noted, we identified several behavioral observations that can be made to help understand and contextualize research findings on student pair programming. In papers that are focused on the topic, these may likely be of similar importance to the above attributes. These include:

(1) Generally, student perceptions of pair programming in the course are: (Enjoy it throughout; Enjoy despite initial hesitations; Begin enjoying but become frustrated as the course continues; Dislike it, despite initial optimism; Dislike it throughout; Seem to have no strong feelings)
(2) When observed pairing, students are often seen: (Pointing to code on screen; Switching roles frequently; Drawing diagrams or gesturing to visualize; Discussing the same problem; Explaining an idea to their partner; Asking questions unrelated to a shared context; Becoming distracted; Researching online for resources; Working on the same problem [with the navigator researching separately / with one following along / independently]; Working on separate problems)
(3) When students change their behavior, it is often: ([specify common time periods in class or semester changes occur]; [specify how the behavior shifts])

5 EXPLORING ADDITIONAL CONTEXTS

This paper makes the case for context through an exploration of pair programming practices that may be critical to study designs in which pair programming is a focus. However, many more contexts exist within the computer science education space which require deeper exploration and context as well. One such example may be flipped classroom practices, which are also commonly cited in the field. We might recommend a similar approach be taken in exploring flipped classroom's context, but with a focus on relevant questions in that area. Such questions may include how often flipped activities occur, what types of flipped activities are used, what portion of the student grade/assessment is based on flipped activity outcomes, how large group sizes are for flipped activities, and how collaborative grouping is determined and structured in the class. Obviously a thorough discussion of this context would warrant its own review and deeper consideration which is out of the scope of this paper. However, this shows the capability of this approach for developing context in areas beyond simply pair programming.

6 CONCLUSIONS AND FUTURE WORK

In this position paper, we have presented the beginning of what we hope is a continued discussion on contextualizing computing education research. This is certainly not a complete representation of context. It is meant as a basis for discussion, but obviously does not and cannot contain all the attributes of a given environment that may be noted in computing education research. As more researchers think about presenting more context for their work, additional significant attributes will likely become apparent, and some presented here may become irrelevant.

Further, the hope is that this work can lead into fidelity of implementation studies like those explored in other disciplines [3, 17, 20, 22] in order to better assess the most critical components of our interventions. In adding context to our papers, we allow for fidelity of implementation to be better assessed in future work.

ACKNOWLEDGMENTS

We thank the students in our CS1 course, for their efforts in pair programming that have motivated our research. We also thank the instructors and assistants in our CS1 course, for their insights and observations regarding pair programming that contributed to our understanding and investigation.

REFERENCES

[1] Kent Beck. 2000. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[2] Briana. Bettin, Jamie L. Berger, Sarah. Larkin, and Leo. Ureel. 2018. Enforcing Positive Pair Programming by Shattering Student Perceptions. Poster presentation at the 2018 Grace Hopper Celebration for Women in Computing.

[3] Maura Borrego, Stephanie Cutler, Michael Prince, Charles Henderson, and Jeffrey E. Froyd. 2013. Fidelity of Implementation of Research-Based Instructional Strategies (RBIS) in Engineering Science Courses. *Journal of Engineering Education* 102, 3 (2013), 394–425. https://doi.org/10.1002/jee.20020

[4] Sallyann Bryant. 2005. Rating expertise in collaborative software development. In *in Proc. PPIG*. PPIG, University of Sussex, Brighton, 19–29.

[5] Jan Chong and Tom Hurlbutt. 2007. The Social Dynamics of Pair Programming. In *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*. IEEE Computer Society, Washington, DC, USA, 354–363. https://doi.org/10.1109/ICSE.2007.87

[6] Katelyn M. Cooper and Sara E. Brownell. 2016. Coming Out in Class: Challenges and Benefits of Active Learning in a Biology Classroom for LGBTQIA Students. *CBE-Life Sciences Education* 15, 3 (2016), ar37. https://doi.org/10.1187/cbe.16-01-0074 PMID: 27543636.

[7] Brian Hanks. 2005. Student Performance in CS1 with Distributed Pair Programming. *SIGCSE Bull.* 37, 3 (June 2005), 316–320. https://doi.org/10.1145/1151954.1067532

[8] Lucas Layman. 2006. Changing Students' Perceptions: An Analysis of the Supplementary Benefits of Collaborative Software Development. In *Proceedings of the 19th Conference on Software Engineering Education & Training (CSEET '06)*. IEEE Computer Society, Washington, DC, USA, 159–166. https://doi.org/10.1109/CSEET.2006.10

[9] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. A Review of Introductory Programming Research 2003–2017. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 342–343. https://doi.org/10.1145/3197091.3205841

[10] Ian McChesney. 2016. Three Years of Student Pair Programming: Action Research Insights and Outcomes. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 84–89. https://doi.org/10.1145/2839509.2844565

[11] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. 2006. Pair Programming Improves Student Retention, Confidence, and Program Quality. *Commun. ACM* 49, 8 (Aug. 2006), 90–95. https://doi.org/10.1145/1145287.1145293

[12] Scott A. Myers. 2012. Students' Perceptions of Classroom Group Work as a Function of Group Member Selection. *Communication Teacher* 26, 1 (2012), 50–64. https://doi.org/10.1080/17404622.2011.625368

[13] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. 2013. Success in Introductory Programming: What Works? *Commun. ACM* 56, 8 (Aug. 2013), 34–36. https://doi.org/10.1145/2492007.2492020

[14] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch. 2009. An empirical study of the effects of personality in pair programming using the five-factor model. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, Washington, DC, USA, 214–225. https://doi.org/10.1109/ESEM.2009.5315997

[15] Allison Scott, Alexis Martin, Frieda McAlear, and Sonia Koshy. 2017. Broadening Participation in Computing: Examining Experiences of Girls of Color. *ACM Inroads* 8, 4 (Oct. 2017), 48–52. https://doi.org/10.1145/3149921

[16] Beth Simon and Brian Hanks. 2008. First-year Students' Impressions of Pair Programming in CS1. *J. Educ. Resour. Comput.* 7, 4, Article 5 (Jan. 2008), 28 pages. https://doi.org/10.1145/1316450.1316455

[17] Marilyne Stains and Trisha Vickrey. 2017. Fidelity of Implementation: An Overlooked Yet Critical Construct to Establish Effectiveness of Evidence-Based Instructional Practices. *CBE life sciences education* 16 (02 2017), rm1. https://doi.org/10.1187/cbe.16-03-0113

[18] Elli J. Theobald, Sarah L. Eddy, Daniel Z. Grunspan, Benjamin L. Wiggins, and Alison J. Crowe. 2017. Student perception of group dynamics predicts individual performance: Comfort and equity matter. *PLOS ONE* 12, 7 (07 2017), 1–16. https://doi.org/10.1371/journal.pone.0181336

[19] Lynda Thomas, Mark Ratcliffe, and Ann Robertson. 2003. Code Warriors and Code-a-phobes: A Study in Attitude and Pair Programming. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, New York, NY, USA, 363–367. https://doi.org/10.1145/611892.612007

[20] Chandra Turpen, Melissa Dancy, and Charles Henderson. 2016. Perceived affordances and constraints regarding instructors' use of Peer Instruction: Implications for promoting instructional change. *Physical Review Physics Education Research* 12 (02 2016). https://doi.org/10.1103/PhysRevPhysEducRes.12.010116

[21] Nanette Veilleux, Rebecca Bates, Cheryl Allendoerfer, Diane Jones, Joyous Crawford, and Tamara Floyd Smith. 2013. The Relationship Between Belonging and Ability in Computer Science. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 65–70. https://doi.org/10.1145/2445196.2445220

[22] Emily M. Walter, Charles R. Henderson, Andrea L. Beach, and Cody T. Williams. 2016. Introducing the Postsecondary Instructional Practices Survey (PIPS): A Concise, Interdisciplinary, and Easy-to-Score Survey. *CBE-Life Sciences Education* 15, 4 (2016), ar53. https://doi.org/10.1187/cbe.15-09-0193 arXiv:https://doi.org/10.1187/cbe.15-09-0193 PMID: 27810868.

[23] Linda L. Werner, Brian Hanks, and Charlie McDowell. 2004. Pair-programming Helps Female Computer Science Students. *J. Educ. Resour. Comput.* 4, 1, Article 4 (March 2004), 8 pages. https://doi.org/10.1145/1060071.1060075

[24] Laurie Williams. 2007. Lessons Learned from Seven Years of Pair Programming at North Carolina State University. *SIGCSE Bull.* 39, 4 (Dec. 2007), 79–83. https://doi.org/10.1145/1345375.1345420

[25] Laurie Williams and Robert Kessler. 2000. Experiments with Industry's "Pair-Programming" Model in the Computer Science Classroom. *IEEE Software - SOFTWARE* 11 (01 2000), 11. https://doi.org/10.1076/csed.11.1.7.3846

[26] Laurie Williams, Lucas Layman, Jason Osborne, and Neha Katira. 2006. Examining the Compatibility of Student Pair Programmers. In *Proceedings of the Conference on AGILE 2006 (AGILE '06)*. IEEE Computer Society, Washington, DC, USA, 411–420. https://doi.org/10.1109/AGILE.2006.25

[27] Krissi Wood, Dale Parsons, Joy Gasson, and Patricia Haden. 2013. It's Never Too Early: Pair Programming in CS1. In *Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136 (ACE '13)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 13–21. http://dl.acm.org/citation.cfm?id=2667199.2667201

[28] Murat Yilmaz, Rory V. OConnor, Ricardo Colomo-Palacios, and Paul Clarke. 2017. An Examination of Personality Traits and How They Impact on Software Development Teams. *Inf. Softw. Technol.* 86, C (June 2017), 101–122. https://doi.org/10.1016/j.infsof.2017.01.005