Improving Interactive Instruction: Faculty Engagement Requires Starting Small and Telling All

BAILEY KACSMAR, University of Waterloo, Canada

Interactive instruction, such as student-centered learning or active learning, is known to benefit student success as well as diversity in computer science. However, there is a persistent and substantial dissonance between research and practice of computer science education techniques. Current research on computer science education, while extensive, sees limited adoption beyond the original researchers. The developed educational technologies can lack sufficient detail for replication or be too specific and require extensive reworking to be employable by other instructors. Furthermore, instructors face barriers to adopting interactive techniques within their classroom due to student reception, resources, and awareness. We argue that the advancement of computer science education, in terms of propagation and sustainability of student-centered teaching, requires guided approaches for incremental instructional changes as opposed to revolutionary pedagogy. This requires the prioritization of lightweight techniques that can fit within existing lecture formats to enable instructors to overcome barriers hindering the adoption of interactive techniques. Furthermore, such techniques and innovations must be documented in the form of computing education research artifacts, building upon the practices of software artifacts.

1 INTRODUCTION

A traditional lecturing style is a common choice for an instructor preparing for a course, but it is not the only one available. Active learning techniques are one way to diverge from traditional lecturing that has been found to improve student engagement and performance [10, 30]. Active learning, generally speaking, encompasses instructional techniques where the students, or learners, are involved in the learning process. Essentially, active learning is a student-centered practice where the students are engaged in doing something besides passive listening. Active learning is not a novel development in educational research, with active learning approaches going back to the early 1900s [32]. For computing education specifically, active learning approaches go back to the 1990s [3]. However, even given this history, it is not unusual for computer science classrooms to employ a traditional instructor-based lecturing approach [12]. This is despite advances in computing education and evidence that lecture-based instruction is a less than optimal teaching technique.

The benefits of student-centered teaching are particularly high during introductory or foundations courses; typically in the first half of degree programs [15, 29, 34]. Furthermore, research targeting the effectiveness of active learning for other science disciplines, such as physics and biology, have found active learning to particularly benefit women and underrepresented minorities [14, 25]. Meanwhile, computer science suffers from attrition rates, accessibility, and diversity issues [5, 20, 38]. As early as 2007, we have case studies on the effectiveness of active learning for supporting diversity in computer science [6]. Yet, there remains a dichotomy between computing education research and practice; where practitioners continue to fall back to conventional lecturing in their classrooms [12, 16–18]. Demographics, institution type, and professor rank have been found to correspond with different rates of usage for student-centered practices [12]. Though few instructors report student-centered learning as their primary teaching strategy, many report using at least one student-centered technique during a course [17] and even small changes towards student-centered active learning can benefit students [22, 23].

Resistance to change, even beneficial change is understandable. Therefore, when considering the incorporation of active learning into computer science classrooms, we need to acknowledge the familiarity of the traditional lecture format. Both instructors and students may feel resistant to changing the format of learning they are used to. Computer science educators want to ensure that specific techniques, which may require additional work on their part, have sufficient support to motivate them to administrators. Instructors may also want assurances that employing these techniques will benefit students and that they can feel confident that their efforts towards implementing techniques will have the desired effects. Research on active learning techniques, for computer science or otherwise, do not necessarily provide clear instruction on how to implement these techniques if modifications are required to satisfy the instructor's course or curriculum specifications. Active learning techniques for computer science also lack the inclusion of clear measures for evaluating their effectiveness. Without clear evaluation metrics instructors are left to guess on the effectiveness based off student evaluations and grades with little intermediate feedback available. Instructors are also left without a path to clearly show students how deviating from the expected instructional framework can actually be beneficial to the students' own goals.

We focus on the accessibility and propagation of interactive computer science education innovations to instructors and the research community. We refer to the past literature as well as lessons from software engineering and other research communities to answer how we can practically and effectively increase the adoption of active learning in computer science classrooms. We propose that a focus on small activities that fit within existing lectures as well as increased documentation in the form of computing education artifacts for all published computing education innovations is best suited to helping instructors overcome barriers they face in terms of time, resources, and student perceptions. Further, through the introduction of artifacts to the computing education community, it will be possible to provide earlier engagement and training to junior researchers and other newcomers to the community.

2 BARRIERS TO INCORPORATING INTERACTIVE LEARNING

Despite an extensive research literature, there remains a gap between theory and practice for computer science education. There are active approaches that can apply to teaching any computer science topic, as well as ones that target specific computer science concepts (such as algorithms [41], artificial intelligence [31, 35], operating systems [22], logic circuits [21]), or introductory programming [27]. Such a dissonance between literature and practice cannot be simply attributed to any one factor [12].

There are a number of factors that impact instructors' motivation. Factors, such as awareness of techniques, time and effort constraints, available training and mentoring, fit with existing practices, time to cover content, and student perceptions or resistance of change have all been identified as potential barriers to the adoption of education innovations [17, 40]. These reasons can be summarized as the risks outweighing the rewards. An instructor who is aware of techniques must invest their time into developing their courses without a clear way to measure their success. Further, the potential cost can be substantial in terms of student performance and student evaluations. End-of-term course evaluations play a large role in instructor decisions to continue using unconventional teaching, likely in part due to concern for student perspectives [16]. Whether the work of researchers is supporting instructors needs and goals has recently come into question [7]. Although designing adoptable innovations falls on researchers, ultimately what it means is design innovations that instructors will be willing and able to use. That is, design innovations that faculty can be convinced to employ.

2.1 Institutional Barriers

Institutional support is beneficial in the propagation of active learning techniques into classrooms. For instance, a university program where more senior mentors are willing and able to help other faculty develop and grow their active learning teaching skills. Despite the benefits of training and other institutional resources, it is a reality that not all instructors have access to such resources. Addressing instructors' concerns with respect to impacts on their career advancement and time requires institutional change such that efforts to improve student success and engagement are not punished. Support for time spent on mentoring must also be rewarded. Without mentoring and guidance it is hard to execute new teaching techniques well and if the execution is done poorly, it will likely fail. Timeslots, labs, and course distribution are also controlled by institutions, and can prove limiting for diversifying instructional techniques.

2.2 Accessibility

University courses are taught with increasing frequency by non-permanent staff. That is, instead of tenured or tenure track faculty many courses are taught by adjuncts, grad students, or those in other 'temporary' instructor roles [11]. Instructors in these roles may be teaching multiple courses a term across more than one institution with no guarantees on teaching the same course from term to term or year to year. Some instructors may be teaching a course while trying to fulfill their degree requirements and preparing for applying to more permanent faculty positions.

While instructors with more permanent positions are able to build up their course repertoire over time, leaving more opportunities to explore novel teaching techniques, it is harder to build reusable material without this stability. Adjuncts and other precariously employed instructors may have to consistently build up new material for each course they teach and be constrained by institutional conventions that do not permit much deviation from lectures. Across instructors there is high variance in terms of their time, freedom, and security. Thus, to benefit students it is important to be mindful of such instructor limitations and focus on developing teaching innovations that bridge these inequities rather than amplifying them.

2.3 Student Perceptions

One of the justifications instructors use for quitting or not trying student-centered interactive learning is concern for negative student evaluations or student push-back [16]. Both students and instructors can experience drawbacks from efforts to employ active learning despite the benefits suggested by the literature. Student perspectives, which can come out through end-of-term evaluations, are used in faculty hiring, promotion, tenure, and salary decisions [1]. These student evaluations disproportionately negatively affect women and minorities, who are also the ones more likely to use unconventional instruction [9, 13]. Previous work on computing education has looked at students' perceptions of engaging with the material when self direction was required of them [19], students' perceptions of team evaluation methods [39], and what factors influence student motivation [33].

In other disciplines, that have investigated student preferences and perceptions, it has been found that active learning is good for students' outcomes, but they do not necessarily like it, that is "Students in the active learning condition reported greater retention of and engagement with the course material but not greater enjoyment when compared to students in the content review condition" [36]. Further, the ultimate valuation of active learning for students can be as simple as the return and that "Any activity, be it active, cooperative or traditional, that directly relates to improving exam performance was the most valued of all" [28]. Thus, student perceptions, and to a degree performance, can be boosted by addressing their concerns and showing how active learning can help them achieve their goals [4, 8].

3 A FOUNDATION IN SMALL TEACHING

The development of computer science specific variants of generic active learning and documented methods to employ them can improve accessibility, use, and sustainability. Despite differences in curriculum or program languages, realistically, all universities have topics and concepts in common that they must teach. There are foundational concepts, not just in introductory courses, but also in more advanced courses that must be taught regardless of other details.

A university that never taught its computer science students about sorting or never introduced the idea of loops or abstraction would be setting their students up for failure. Thus, we oppose past concerns that developing concept specific techniques can limit usage across universities [40].

We advocate for the use of smaller interventions from *Small Teaching* as a way to slowly integrate interactions into a course from the beginning [23]. The goals of small teaching include low time requirements (in terms of preparation, execution, and evaluation) while still being provably beneficial to student engagement and learning. That is, "quick small active teaching" is a way to help get both instructors and students adjust to the changes in interaction levels in their courses. Large techniques, with sufficient complexity or targeted at courses rather than concepts, require computing education researchers to provide details on how to adapt them before they can be useful to other instructors [40]. We will further discuss a solution to ensuring these details exist in Section 4.

3.1 Small Teaching

In this work, we use the term 'small teaching' to encompass activities such as those highlighted by Lang [23]. These activities include ones where the activity takes between five and ten minutes, may be a one-time intervention in the course, and only requires a small modification to the student experience or course design. We also include activities in this work towards the medium or large side, but we emphasize that such activities would require a greater effort on the part of an instructor, and even further documentation efforts on the part of researchers (see artifacts in Section 4).

An example of an activity that can be used as 'small teaching' is the one-minute thesis. A 'one-minute thesis' activity can be done during existing lecture time and only requires the preparation of a prompt on the part of the instructor. The prompt is generally open ended and has the students write everything they can in response to the prompt, but within one minute. An instructor can then review these responses to: gauge student understanding, identify gaps, and then modify their instruction going forward to address any gaps or issues in understanding. As it only takes one minute of writing, it requires low effort on the part of the student while still generating engagement and providing feedback.

3.2 Supporting Instructors

To demonstrate the importance of starting small for instructors, we invite readers to consider the following educational experiences from their past. Before anyone first writes an essay or a research paper, they begin with sentences and paragraphs. Before releasing an application to users, the programmers first learn to build small projects that provide them with the building blocks they will need to explore and develop larger systems. Computing education instructors at the university level are generally PhDs with research training, which does not require training as a teacher. Furthermore, instructors at different career points each face barriers to adopting teaching innovations. Recall that course evaluations are used in hiring and promotional decisions; affecting sessionals, adjuncts and tenure-track instructors. Tenure-track research professors may find taking time away from their regular research to learn about teaching innovations may be too big of a time investment to risk the role research plays in their promotions. Thus, encouraging faculty adoption of teaching innovations requires balancing the risk and time investment of change against the reward. To tip the scales towards change, education innovations need to recognize instructors as being computing education students with time limitations and limited access to external teaching training. If instructors are provided with small guided techniques to explore, they can develop their own skills much as students would by beginning with small exercises and growing from there. Computing education instructors deserve the same opportunities to grow their skills as their students.

3.3 Supporting Students

It would be remiss to forget that grades do still matter to students and that there are legitimate reasons for that. In a university or similar facility, in general, students must receive a grade at the end of the course. These students may require certain grades to satisfy degree requirements or to maintain funding for their studies (e.g., scholarships or student loans). Some students may be working part-time to fund their studies and only have a limited amount of time to dedicate to the material in the course (in addition to their other courses). Such students may only be able to spend enough time to ensure they continue to meet their degree requirements. These are all realities that may outweigh a student's motivation to master material and should be considered when determining what strategies to incorporate.

Deviation from what students expect from their classes can result in push back and resistance to the change. Resistance can be particularly strong if change is introduced part way through the term [4]. Fortunately, student resistance is not nearly as common as instructors may fear or perceive it to be [4]. Furthermore, resistance can be assessed, and it can be mitigated. To counter students' resistance to active learning it is important to start off on the first day with interaction and an explanation of what students can expect. According to work from Chasteen on helping students engage with active learning, other than starting from the first day, there are four student concerns you should ensure you address [4]. First, students want to know how the class works. This includes grading schemes and course content, but also how class time will be spent. Students want to know how to get a good grade in the course and correspondingly to understand how the instructor's style of teaching the course will help them get that grade. Finally, when students are asked to contribute and interact in the course, they want to know whether their contribution will be valued.

The advantages of countering potential student resistance include: established effectiveness of small teaching that can be shared with students, low preparation time, low activity time, and can be included from day one of a course. Instructors wishing to use small techniques can present the student benefits on the first day when explaining the course syllabus and other information about how the course is run. They can begin the course with a small activity, for instance, a prompt of "What do you think you will learn in this course"? The instructor can demonstrate student contributions matter by addressing these responses as well as whether some suggestions can be incorporated into the course now that the instructor knows there is interest. Such small first day activities can set the tone for the course, show students the instructor is open to their input, and open the door to more complex activities later in the term without being completely unexpected. Motivation can be given to students by explaining to them that there is evidence that doing these activities will help their grades, but ultimately students will be more responsive to a direct connection to their grades. A direct benefit can come in the form of giving students bonus grades for participating in these activities [4, 36].

4 ARTIFACTS FOR ACCESSIBILITY

In an ideal world, there would be the creation of a collective database of small easily included active learning techniques available to instructors. While some high school instructors may have sources via their school divisions as well as online resources [24], this is not necessarily the case for university instructors, who additionally may lack training as educators. Across the globe countless instructors are teaching similar courses with similar material. It is possible for precise activities to be described and detailed in terms of specific course content and still be useful across numerous university's courses. Barring such a database, there can at least be an increase in innovation documentation in computing education research focused on reusability. In terms of human efforts, the creation and review of artifacts shifts time and effort away from instructors, but onto researchers. Researchers must put in additional effort to ensure that their innovation is accessible to others for replication. However, this extra effort also benefits researchers by facilitating replication studies and the information needed to build upon colleagues' work, just as in other areas of computer science.

4.1 Research Artifacts in Computer Science

In recent years there has been a rising prioritization of making computing research more reproducible and accessible through recognizing and reviewing research artifacts. Research artifacts build on the idea of software artifacts which can include mock-ups and other components generated during the design process of a software system as well as documentation and demos for the software itself. Separate artifact reviews for accepted papers or for replication studies have been added in software engineering at the International Conference on Software Engineering (ICSE)[1], in operating systems at Operating Systems Design and Implementation (OSDI)[2], in security at USENIX Security[3], and in privacy at the Privacy Enhancing Technologies Symposium (PETS)[4]. Across the above research communities, we see a breadth of artifacts. Types of artifacts evaluated by at least one of the above venues include: software, data sets, survey results, test suites, mechanized proofs, source code, scripts for data processing or simulations, formal specifications, build environments, hardware, and frameworks (tools and services illustrating new approaches usable in different contexts).

4.2 Computer Science Education Artifacts

Within the computing education community, SIGCSE-TS includes a conference track for Nifty Assignments[5] that requests computer science assignments to be submitted as well as experience report submissions. Unlike research paper artifacts, however, these are distinct from computer science education papers accepted to the conference. Thus, they serve an important, but different contribution to the computing education community than research artifacts. In our call for computing education artifacts we are focusing on the need to document education innovations and studies that can be used by instructors and other researchers to improve the community at large.

Artifact Specifications. Similar to other computer science artifacts, computing education artifacts are not limited to just one form. Artifacts can include: datasets, interactive activities, software applications, activity design strategies, and teaching methods, to name a few. The priority in calling for artifacts is not to limit what can be an artifact. Instead, our call for artifacts is to advocate for usability and accessibility by rewarding the documentation and disclosure of components discussed and developed for a teaching innovation paper. For researchers and educators that hope to have education innovations adopted by instructors, we highlight the following requirements: preparation or design time, in class duration, student motivation, computing or other resource requirements, and general documentation in terms of instructions for use or fidelity of the implementation.

When instructors other than the original researchers attempt to use a technique in their classrooms a specific measurement for successful use is beneficial. This measurement targets the fidelity of the implementation and is used to validate that the instructor has delivered the technique as it was intended [37, 40]. Without such validation metrics, there is less reassurance that the motivating goals (e.g., in terms of student success), can and will be met.

Further, artifacts must be hosted in a publicly accessible format and location to be useful to both researchers and instructors. During the submission stage, the current solution in other venues in computer science (ICSE, OSDI, USENIX

[1]https://conf.researchr.org/track/icse-2021/icse-2021-Artifact-Evaluation#Call-For-Artifact-Submissions
[2]https://www.usenix.org/conference/osdi20/call-for-artifacts
[3]https://www.usenix.org/conference/usenixsecurity21/artifact-evaluation-information
[4]https://petsymposium.org/artifacts.php
[5]http://sigcse2022.sigcse.org/authors/nifty/

Security, PETS) is to have the authors host their artifact on publicly accessible author chosen systems such as GitHub or institutional websites. Depending on the size of the venue soliciting these artifacts, once accepted the venue can either reference links to these author-hosted artifacts alongside the published papers or host the artifacts on a conference server. Hosting the artifacts is obviously more resource intensive but does ensure a stronger sense of future accessibility.

Artifact Reviewers. To have artifacts reviewed requires an additional reviewing committee. Such a committee is necessary to ensure potentially already over-strained program committees are not further overextended. Artifact committees at other venues in computer science (ICSE, OSDI, USENIX Security, PETS), have invited more junior researchers to these committees, including having senior PhD students and post-docs as co-chairs. Junior researchers are an excellent choice for evaluating artifacts for usability as their levels of expertise may have greater similarity to instructors new to using teaching innovations than more senior researchers. Thus, the junior researchers may be able to identify gaps that would inhibit reusability that more senior researchers would fill in using knowledge from their own experiences. If the junior researchers find an innovation inaccessible, then it may not be ready for instructors who may have even less experience or familiarity with computer science education innovations.

4.3 CSE Research Community Engagement

The geographic diversity of the CSE research community, specifically that of SIGCSE, finds limited representation outside of North America [2]. This limitation is also found with respect to junior scholar participation as evaluated through their involvement in doctoral consortia. While Doctoral consortia (e.g., at ITiCSE and ICER) provide graduate students with opportunities to share their work, they previously required in-person attendance which may have contributed to the primarily North American based participation [2, 26]. Unlike doctoral consortia, artifact reviewing can be done entirely through online participation without loss of engagement. Thus, our proposal while focused on propagation of teaching innovations to instructors and other researchers, additionally provides a non-geographic dependent engagement opportunity for researchers new to the CSE research community to participate. Further, artifact reviewing provided junior scholars with additional networking opportunities with colleagues as well as academic training in the form of reviewer experience. Encouraging junior researchers participation on artifact committees benefits the community by having these artifacts evaluated for usability, and providing training and experience to junior researchers.

5 CONCLUSIONS

An all or nothing treatment of active learning, as a sort of pedagogical revolution, is inaccessible to the majority of instructors, and thus, cannot benefit students. Even less revolutionary ideas, such as a specific way to teach a specific concept with a specific technology can still have a lot of overhead that can be overwhelming or inaccessible. Educational innovations must add clear instructions on how to adapt them, if necessary, to different contexts such as class size, time, and content. Instructors must be provided with detailed instructions and measures for integrating and evaluating the inclusion of novel teaching techniques in their classrooms. Without support, and given the potential consequences from negative student evaluations, it is understandable that instructors fall back to the safety of traditional lecturing.

In this short paper, we highlighted the need for a prioritization of computing education artifacts, and in particular for small teaching techniques, to support instructors in developing their own knowledge and understanding without requiring them to work through the bountiful set of literature. By designing education innovations with these corresponding prioritizations, that include detailed instructions for teaching specific computer science topics, we can improve instructor adoption. Conferences and other research venues have the power to support students by driving change and improving the accessibility of active learning techniques for computer science educators through rewarding researchers' efforts in producing reusable teaching innovations. Change requiring institutional support is a long-term process, however, through starting small and telling all, in the form of small teaching and artifacts, the computing education community can begin to address barriers to adoption with greater immediacy.

REFERENCES

[1] Susan A Basow and Julie L Martin. 2012. *Bias in student evaluations. In M. E. Kite (Ed.), Effective evaluation of teaching: A guide for faculty and administrators.* Society for the Teaching of Psychology, 40–49.

[2] Brett A. Becker, Amber Settle, Andrew Luxton-Reilly, Briana B. Morrison, and Cary Laxer. 2021. *Expanding Opportunities: Assessing and Addressing Geographic Diversity at the SIGCSE Technical Symposium.* ACM, New York, NY, USA, 281–287.

[3] Stefan Biffl and G Thomas. 1998. Preparing students for industrial teamwork: a seasoned software engineering curriculum. *IEEE Proceedings-Software* 145, 1 (1998), 1–11.

[4] Stephanie Chasteen. 2017. How do I help students engage productively in active learning classrooms. *posted to https://www. physport. org/recommendations/Entry. cfm* (2017).

[5] J. McGrath Cohoon. 2001. Toward Improving Female Retention in the Computer Science Major. *Commun. ACM* 44, 5 (may 2001), 108–114.

[6] James P. Cohoon. 2007. An Introductory Course Format for Promoting Diversity and Retention. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA) *(SIGCSE '07)*. ACM, New York, NY, USA, 395–399.

[7] Paul Denny, Brett A. Becker, Michelle Craig, Greg Wilson, and Piotr Banaszkiewicz. 2019. Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. ACM, New York, NY, USA, 259–267.

[8] Rutwa Engineer, Ayesha Naeem Syeda, and Bogdan Simion. 2021. A Qualitative Study of Group Work and Participation Dynamics in a CS2 Active Learning Environment. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Virtual Event, Germany) *(ITiCSE '21)*. ACM, New York, NY, USA, 25–31.

[9] Yanan Fan, Laura J Shepherd, Eve Slavich, David Waters, M Stone, R Abel, and Emma L Johnston. 2019. Gender and cultural bias in student evaluations: Why representation matters. *PloS one* 14, 2 (2019), e0209749.

[10] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences* 111, 23 (2014), 8410–8415.

[11] Judith M Gappa. 2008. Today's majority: Faculty outside the tenure system. *Change: The Magazine of Higher Learning* 40, 4 (2008), 50–54.

[12] Scott Grissom, Sue Fitzgerald, Renée McCauley, and Laurie Murphy. 2017. Exposed! CS Faculty Caught Lecturing in Public: A Survey of Instructional Practices. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. ACM, New York, NY, USA, 261–266.

[13] Scott Grissom, Renée Mccauley, and Laurie Murphy. 2017. How Student Centered is the Computer Science Classroom? A Survey of College Faculty. *ACM Trans. Comput. Educ.* 18, 1, Article 5 (Nov. 2017), 27 pages.

[14] David C Haak, Janneke HilleRisLambers, Emile Pitre, and Scott Freeman. 2011. Increased structure and active learning reduce the achievement gap in introductory biology. *Science* 332, 6034 (2011), 1213–1216.

[15] John P Holdren and Eric Lander. 2012. Engage to excel: Producing one million additional college graduates with degrees in science, technology, engineering, and mathematics. *President's Council of Advisors on Science and Technology* (2012).

[16] Christopher Lynnly Hovey and Lecia Barker. 2020. Faculty Adoption of CS Education Innovations: Exploring Continued Use. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. ACM, New York, NY, USA, 570–576.

[17] Christopher Lynnly Hovey, Lecia Barker, and Margaret Luebs. 2019. Frequency of Instructor- and Student-Centered Teaching Practices in Introductory CS Courses. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 599–605.

[18] Christopher Lynnly Hovey, Kathleen J. Lehman, and Tiffani Riggers-Piehl. 2020. Linking Faculty Attitudes to Pedagogical Choices: Student-Centered Teaching in Introductory Computing Classes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. ACM, New York, NY, USA, 584–590.

[19] Ville Isomöttönen and Ville Tirronen. 2013. Teaching Programming by Emphasizing Self-Direction: How Did Students React to the Active Role Required of Them? *ACM Trans. Comput. Educ.* 13, 2, Article 6 (jul 2013), 21 pages.

[20] Amanpreet Kapoor and Christina Gardner-McCune. 2018. Considerations for Switching: Exploring Factors behind CS Students' Desire to Leave a CS Major. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018)*. ACM, New York, NY, USA, 290–295.

[21] Ville Karavirta, Rolf Lindén, Einari Kurvinen, and Mikko-Jussi Laakso. 2016. *Interactive Exercises for Teaching Logic Circuits.* ACM, New York, NY, USA, 101–105.

[22] Michael S. Kirkpatrick and Samantha Prins. 2015. Using the Readiness Assurance Process and Metacognition in an Operating Systems Course. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. ACM, New York, NY, USA, 183–188.

[23] James M Lang. 2016. *Small teaching: Everyday lessons from the science of learning.* John Wiley & Sons.

[24] Mackenzie Leake and Colleen M Lewis. 2017. Recommendations for designing CS resource sharing sites for all teachers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.* 357–362.

[25] Mercedes Lorenzo, Catherine H Crouch, and Eric Mazur. 2006. Reducing the gender gap in the physics classroom. *American Journal of Physics* 74, 2 (2006), 118–122.

[26] Stephanie Lunn, Maíra Marques Samary, and Alan Peterfreund. 2021. *Where is Computer Science Education Research Happening?* ACM, New York, NY, USA, 288–294.

[27] 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. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus). ACM, New York, NY, USA, 55–106.

[28] Patricia L Machemer and Pat Crawford. 2007. Student perceptions of active learning in a large cross-disciplinary classroom. *Active learning in higher education* 8, 1 (2007), 9–30.

[29] Dan MacIsaac. 2015. What research says about effective instruction in undergraduate science and engineering (2015). *The Physics Teacher* 53, 3 (2015), 190–190.

[30] Joel Michael. 2006. Where's the evidence that active learning works? *Advances in physiology education* 30, 4 (2006), 159–167.

[31] Keith O'Hara, Douglas Blank, and James Marshall. 2015. Computational notebooks for AI education. In *The Twenty-Eighth International Flairs Conference.* Florida Artificial Intelligence Research Society Conference.

[32] PK Rangachari. 2007. Back to the future? Active learning of medical physiology in the 1900s. *Advances in physiology education* 31, 4 (2007), 283–287.

[33] Merilin Säde, Reelika Suviste, Piret Luik, Eno Tõnisson, and Marina Lepp. 2019. Factors That Influence Students' Motivation and Perception of Studying Computer Science. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 873–878.

[34] Susan Singer and Karl A Smith. 2013. Discipline-based education research: Understanding and improving learning in undergraduate science and engineering. *Journal of Engineering Education* 102, 4 (2013), 468–471.

[35] Sameer Singh and Sebastian Riedel. 2016. Creating interactive and visual educational resources for ai. In *Thirtieth AAAI Conference on Artificial Intelligence.*

[36] C Veronica Smith and LeeAnn Cardaciotto. 2011. Is active learning like broccoli? Student perceptions of active learning in large lecture classes. *Journal of the Scholarship of Teaching and Learning* 11, 1 (2011), 53–61.

[37] 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, 1 (2017), rm1.

[38] C Stephenson, A Derbenwick Miller, C Alvarado, L Barker, V Barr, T Camp, C Frieze, C Lewis, E Cannon Mindell, L Limbird, et al. 2018. Retention in Computer Science Undergraduate Programs in the US: Data Challenges and Promising Interventions. *New York, NY. ACM* (2018).

[39] Anya Tafliovich, Andrew Petersen, and Jennifer Campbell. 2016. Evaluating Student Teams: Do Educators Know What Students Think?. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. ACM, New York, NY, USA, 181–186.

[40] Cynthia Taylor, Jaime Spacco, David P. Bunde, Zack Butler, Heather Bort, Christopher Lynnly Hovey, Francesco Maiorana, and Thomas Zeume. 2018. Propagating the Adoption of CS Educational Innovations. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus). ACM, New York, NY, USA, 217–235.

[41] J. Ángel Velázquez-Iturbide. 2013. An Experimental Method for the Active Learning of Greedy Algorithms. *ACM Trans. Comput. Educ.* 13, 4, Article 18 (nov 2013), 23 pages.