2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52130

NSF HSI:ELPSE Project: Early Design and Pilot Evidence from an Initiative Aimed at Increasing Engagement in Foundational Computer Science Classes

Dr. Shanon Marie Reckinger, University of Illinois at Chicago

Shanon Reckinger is a Clinical Associate Professor in the department of Computer Science (CS) at the University of Illinois at Chicago.

Gonzalo Bello, The University of Illinois at Chicago
Prof. Robert Sloan
Mitchell D. Theys, University of Illinois at Chicago
Scott J. Reckinger, University of Illinois Chicago

Scott J. Reckinger is a Clinical Associate Professor in the Department of Computer Science at the University of Illinois Chicago. He earned a PhD in Mechanical Engineering at the University of Colorado Boulder. His research areas within computational fluid dynamics include numerical methods for simulations of fluid instabilities, turbulence modeling, and parameterizations for ocean models. He has also published papers in computer science and engineering education, specifically on oral proficiency exams (both synchronous and asynchronous), two-stage exams, project-based learning, and active learning methods.

Baker E Franke, The University of Illinois at Chicago

NSF HSI:ELPSE Project: Early Design and Pilot Evidence from an Initiative Aimed at Increasing Engagement in Foundational Computer Science Classes

Overview

Student engagement and motivation are lacking in our high-enrollment Computer Science (CS) core programming classes, and attrition rates are too high. Teacher preparation, satisfaction, and collaboration also need improvement. We are beginning a five year project to redesign our four CS core programming courses: traditional CS 1 and CS 2 courses, a home-grown course entitled *Programming Practicum* that has CS 2 as a prerequisite (CS PP), and Data Structures (CS DS). Our redesign focuses on student-centered, research-based practices. Our program has three project objectives:

1. Create a distinctive program that attracts and empowers a large pool of emerging computing talent in Chicago.
2. Foster a culture change in the programming core sequence for both current and future instructors and students.
3. Improve retention for all students, and close retention gaps for students from groups where DFW rates are disproportionately high.

We will achieve these goals through three activities: (1) Revamped Programming Core, (2) Collaborative Teaching Model, and (3) Igniting Minds Program. The Revamped Programming Core will align our (i) curriculum, (ii) student learning supports, and (iii) teaching practices across sections and across semesters. The Collaborative Teaching Model will transition our programming core from one or two instructors teaching sections with hundreds of students to many dedicated instructors teaching sections with tens of students through a collaborative, productive, and responsive teaching model. For our program, this will increase the number of full time teaching faculty assigned to the programming core by 3. The Igniting Minds Program will provide optional, for-credit curriculum and workshops for students with less programming experience needing extra support in the development of crucial skills necessary for success with programming and broader CS studies.

Background and Early Institutional Evidence

Our CS department has seen rapid growth over the past decade, reflecting broader enrollment trends in North America [6]. Faculty size has nearly tripled to meet demand. Meanwhile, generative AI tools capable of solving programming problems have raised concerns about deepening learning gaps [4].

Preliminary data show that smaller class sizes in our programming core correlate with lower DFW rates, as shown in Figure 1. The small enrollment sections of Spring 2024 (marked in orange) are outliers to this trend. We speculate that the high DFW rates for these classes are related to cumulative negative effects of generative AI tools in this introductory course. These sections, which immediately followed the large enrollment sections of Fall 2023, experienced high rates of academic misconduct and significantly lower exam scores, largely driven by

overreliance on and misuse of AI tools. We also found that our DFW rates in our programming core are worse for students historically underrepresented in computing.

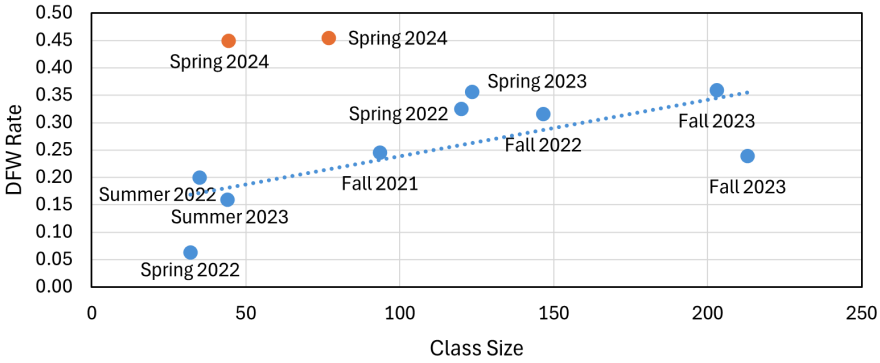

Figure 1 - DFW rates by class size for our CS 1.

This trend follows what has been found in other disciplines with "gateway" classes. For example, large-enrollment gateway courses in Physics [1], Chemistry [2], and Calculus [3] reduced DFW rates by decreasing class sizes, which in turn enabled the incorporation of active and peer learning strategies combined with hierarchical teacher models. CS has yet to adopt the long-established practices of our STEM peers, who have managed large enrollments for decades due to their history of offering service courses to other colleges.

We have experimented with running small class sizes in our program. Figure 2 shows feedback from students who were enrolled in large enrollment vs. small enrollment versions of our CS1 course. This is also broken down by students with some prior programming experience (PPE) and no PPE, as well as high parental education level (PEL) and low parental education level. Low-PEL students reported greater appreciation for the class size, environment, and lab sessions compared to high-PEL students, and the increase in the student-reported appreciation when comparing small enrollment to large enrollment sections is larger for low-PEL students than high-PEL students. Similarly, students with no-PPE reported higher appreciation for the class size compared to some-PPE with a greater increase when comparing small-enrollment to large-enrollment sections. The PPE trends are more mixed on student reported enjoyment of the class and lab environments.

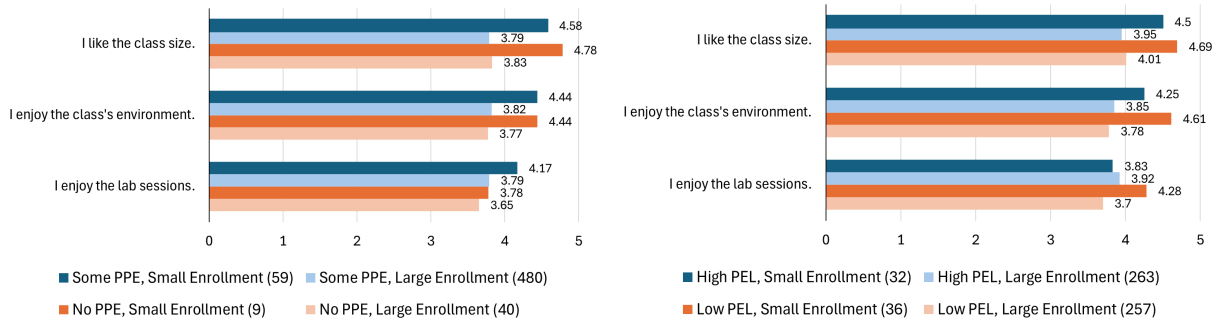

Figure 2 - Student-reported satisfaction of large enrollment vs small enrollment sections. (Note: PPE = prior programming experience; PEL = parental education level.)

Some of our small sections have run in an integrated lab/lecture format. NC State's CS department has been running their CS1 course in an integrated lab/lecture format since 2007. When first implemented and compared against the large lecture format, they found students learning, satisfaction, and engagement improved in the small integrated lab/lecture sections [7]. Our work will build on this exciting body of evidence. We will investigate the effectiveness of small integrated lab/lectures as a replacement for large enrollment lectures with separate labs.

Preliminary Findings

Preliminary results from our CS 1 courses demonstrate that smaller, more personalized learning environments are associated with significantly lower DFW rates, with the most dramatic improvements seen among Black and Hispanic students. DFW rates from Black and Hispanic students are ~10-25% higher than those of their White and Asian peers in our large enrollment CS 1 class. However, in our small integrated lab/lecture sections, the DFW rates between the two groups are statistically indistinguishable. Additionally, students report higher enjoyment of the class environment and collaboration with peers in the smaller classes. By combining the benefits of reduced class sizes with the strength of collaborative teaching, the result is a scalable model that prioritizes equity, engagement, and long-term success. The new model emphasizes alignment of curriculum, teaching practices, and student support, and replaces large lectures with integrated, 30-student sections that combine lecture and lab. These sections are led not by isolated instructors but by teams of experienced full-time faculty who collaborate across the core. This team-based approach ensures consistency, supports innovation, and creates a more coherent and responsive learning environment for all students.

A pilot small-enrollment, integrated lecture-lab CS course is being offered now (Spring 2026). A central research question asks how factors like attentiveness, participation, peer interaction, teacher support, and curriculum impact student satisfaction and achievement. Evaluation includes participant surveys and classroom observations by senior faculty using an adapted Classroom Observation Protocol for Undergraduate STEM (COPUS) protocol for CS [5].

Theoretical Framework

Our proposed research plan is grounded in constructivist learning theory [8, 9], particularly active and collaborative learning. This approach emphasizes that students construct knowledge through engagement rather than passively receiving information. The Revamped Programming Core aims to lower student-to-teacher ratios, allowing for richer, more effective pedagogy. UIC's diverse student body enters with varied educational backgrounds, making a one-size-fits-all starting point ineffective. This challenge, common at large public institutions, underscores the broader relevance of our work.

We expect engagement and collaboration to increase as class sizes shrink. Over the past 15 years, UIC-CS instructors have applied numerous research-based strategies in large lectures including polling software, think-pair-share, and POGIL worksheets among them. While polling is widely used, it alone has not consistently improved outcomes [10], and its impact may be waning due to the shift to mobile-only platforms during the pandemic. Therefore, despite our existing core curriculum heavily utilizing polling systems, we still expect engagement and collaboration to increase after implementing the Revamped Programming Core.

Our work also draws on Finn et al. (2003) [11], who argue that academic achievement is influenced by both academic and social engagement. Academic engagement includes behaviors like class participation, while social engagement involves interactions with peers and instructors. In line with CS education research, we refer to these as engagement and collaboration, respectively. Wang et al. (2022) have done similar work where they looked at engagement and collaboration in introductory business classes [12].

The Revamped Programming Core aligns with this framework by promoting smaller classes run only by qualified instructors, where students are more visible and, thus, more likely to engage and collaborate. In contrast, large lecture classes allow students to remain anonymous, which, in our experience, reduces engagement, collaboration, and even attendance. These courses often use lecture capture, which, while popular with students, has been shown to increase absences and delay learning [13].

Our core priority is straightforward: get students into the classroom. Without attendance, engagement and collaboration, the foundation of our approach, cannot occur.

Study Design

We have completed our IRB approval process and moved into our new building. We have also scheduled a Pilot Study which has launched in Spring of 2026. We will be piloting our new course format in a specialized section of our data structures class. This will allow us to better understand scheduling and logistics, to experiment with pedagogy that we have brainstormed for our curriculum frameworks, and to test out our survey process. The instructor of the pilot study course is external to the research team and will be able to provide unbiased feedback to the team. We will meet with this instructor several times throughout the semester. The teaching model and curriculum frameworks will not be developed yet, so this pilot study is to help us better prepare for the rollout.

In Year 1, we will be developing the core frameworks, teaching model, and Igniting Minds curriculum. During this time, we will be collecting benchmark data in all four courses that use our existing curriculum. This includes collecting survey data from students and teachers. We will recruit students from the four core programming classes that use the existing programming core curriculum during Year 1. We will recruit teachers who have taught any of the four core programming classes over the past 7 years. During this time, we can, if needed, continue to run

small-scale experimental pilot sections to further hone the logistics of these sections. Each semester, we will collect student grades. Additionally, we will administer the validated Data Structures Concept Inventory test to our CS DS students during all years, which is a regular practice in our program. Also, we will conduct classroom observations during this time and, in conjunction, collect data using our Classroom Session Engagement Survey. Unlike the full survey, the Classroom Session Engagement Survey has students reflect specifically about the classroom activity they just engaged in.

We had planned to roll out our first small sections in Year 2. However, that was when our project was starting in the Fall term. Our project start date was pushed back one semester, so we decided to push our timeline up by one semester. The main reason for this is that our enrollments for CS 1 are too small in the Spring to run large and small sections simultaneously. Therefore, we opted to move our timeline up by a semester rather than back a semester. Therefore, the new CS 1 will be offered for student enrollment in Semester 2 of Year 1. We will offer students to enroll in our existing CS 1 course or choose to enroll in our new CS 1 course. This will form two groups for comparison using *two independent groups test* quantitative methods. We will continue to collect survey data from both students and teachers in our existing curriculum, as well as the other items discussed in Year 1.

In Years 3-5, we will continue the same rollout process. For example, in Year 2.5, CS 1 will be run entirely with Revamped Programming Curriculum and CS 2 will be offered with both the existing CS 2 course and the new CS 2 course. Similar to CS 1, this will give us two CS 2 groups to compare. However, another purpose of this rollout is to allow for a secondary analysis using *repeated measures*. The repeated measures would include students who were able to take some of the programming core classes with the Revamped Curriculum and some with the existing curriculum. The rollout would also allow us to incorporate counterbalancing, which strengthens analysis. For example, some students may take CS 1 using the existing curriculum and then CS 2 using the Revamped Curriculum. In other cases, some students may take CS 1 using the Revamped Curriculum and then CS 2 using the existing curriculum. Due to large enrollment of students, this should happen through natural student enrollment and will not be controlled or randomized.

Acknowledgement

The National Science Foundation supported this work through the Hispanic-Serving Institutions: Enriching Learning, Programs, and Student Experiences (HSI:ELPSE) program under Award No. 2524488. The opinions, findings, conclusions, or recommendations presented in this material are solely those of the author(s) and do not necessarily represent the views of the National Science Foundation.

References

[1] J. L. Alzen, L. Langdon, and V. Otero, "The Learning Assistant model and DFW rates in introductory physics courses," in *Proc. Physics Educ. Res. Conf. (PERC)*, Cincinnati, OH, USA, 2017, pp. 36–39.

[2] J. Mutanyatta-Comar and S. R. Mooring, "Evaluation of a Peer-Led Team Learning–Flipped Classroom reform in large enrollment organic chemistry courses," in *ACS Symp. Ser.*, vol. 1341, 2019, pp. 145–157, doi: 10.1021/bk-2019-1341.ch011.

[3] P. R. Norton, W. Bridges, and K. High, "Impact of course policy changes on Calculus I DFW rates," *J. STEM Educ.: Innov. Res.*, vol. 19, no. 1, Mar. 2018. [Online]. Available: https://www.jstem.org/jstem/index.php/JSTEM/article/view/2180

[4] J. Prather *et al*., "The widening gap: The benefits and harms of generative AI for novice programmers," *arXiv* preprint arXiv:2405.17739, 2024. [Online]. Available: https://arxiv.org/abs/2405.17739

[5] M. K. Smith, F. H. M. Jones, S. L. Gilbert, and C. E. Wieman, "The Classroom Observation Protocol for Undergraduate STEM (COPUS): A new instrument to characterize university STEM classroom practices," *CBE—Life Sci. Educ.*, vol. 12, no. 4, pp. 618–627, 2013, doi: 10.1187/cbe.13-08-0154.

[6] S. Zweben and B. Bizot, "Taulbee Survey: CS enrollment grows at all degree levels with increased gender diversity," *Comput. Res. News*, vol. 34, no. 5, pp. 2–82, 2021.

[7] K. E. Boyer, R. S. Dwight, C. S. Miller, C. D. Raubenheimer, M. F. Stallmann, and M. A. Vouk, "A case for smaller class size with integrated lab for introductory computer science," in *Proc. 38th SIGCSE Tech. Symp. Comput. Sci. Educ. (SIGCSE '07)*, New York, NY, USA, 2007, pp. 341–345, doi: 10.1145/1227310.1227430.

[8] M. Ben-Ari, "Constructivism in computer science education," in *Proc. 29th SIGCSE Tech. Symp. Comput. Sci. Educ. (SIGCSE '98)*, 1998, pp. 257–261, doi: 10.1145/273133.274308.

[9] M. Ben-Ari, "Constructivism in computer science education," *J. Comput. Math. Sci. Teach.*, vol. 20, no. 1, pp. 45–73, 2001.

[10] D. J. Weiss, P. McGuire, W. Clouse, and R. Sandoval, "Clickers are not enough," *J. Coll. Sci. Teach.*, vol. 49, no. 3, pp. 58–65, 2020.

[11] J. D. Finn, G. M. Pannozzo, and C. M. Achilles, "The 'why's' of class size: Student behavior in small classes," *Rev. Educ. Res.*, vol. 73, no. 3, pp. 321–368, 2003.

[12] L. Wang and L. Calvano, “Class size, student behaviors and educational outcomes,” *Organ. Manag. J.*, vol. 19, no. 4, pp. 126–142, 2022.

[13] S. Voelkel, A. Bates, T. Gleave, C. Larsen, E. J. Stollar, G. Wattret, and L. V. Mello, “Lecture capture affects student learning behaviour,” *FEBS Open Bio*, vol. 13, no. 2, pp. 217–232, 2023, doi: 10.1002/2211-5463.13548.