

**Putting the CS in Computing Education Research**

In April of 2013, San Jose State University announced an expansion of their pilot MOOC courses with EdX based on a successful offering of EE98 Circuits Analysis. In July of 2013, San Jose State University suspended their MOOC project with EdX when half of the students taking the courses failed their final exams.

In August of 2014, code.org released a K-6 curriculum to the world that had been created a few months earlier, not having been tested rigorously in elementary school classrooms.

What do these two events have in common? Computer scientists identified a critical need in computer science education (and education in general) and developed something new to fill that need, released it, and scaled it without rigorous, scientific experiments to understand in what circumstances they are appropriate. Those involved have the best of intentions, working to create a solution in an area with far too little research. A compelling need for greater access to computing education, coupled with a dire shortage of well-supported computing education researchers, has led to deployments that come before research. High-profile failures hurt computer science’s credibility in education, which in turn hurts our future students. This imbalance between the demand and supply and the chasm between computer science and education creates an opportunity some forward-thinking departments.

If computer science wants to be a leader rather than a spectator in this field, computer science departments at PhD-granting institutions must hire faculty in Computing Education Research (CER) to transform the face of education – in undergraduate CS teaching, K-12 CS education, and education in general.

**Finding a place**

As with any interdisciplinary field, we must explore in which department should this research be performed, and in what role should this researcher be hired? To understand where computing education research belongs, we need to understand what computing education research is. I divided it roughly into two categories.

1. How students learn computing concepts, whether that is K-12 or college, how their prior experiences (gender, ethnicity, socioeconomic background, geographic region, generation) influence how they learn, and how those findings influence the ways we should teach.
2. What interfaces, languages, classroom techniques and themes are useful for teaching CS concepts.

The appropriate department depends on the particular research questions being asked. I am not advocating that every department should hire in CER, nor that all CER research should occur in CS departments. However, the biggest opportunity lies in hiring leaders who will assemble an interdisciplinary team of education faculty, CS faculty, CS instructors, and graduate students to make transformative contributions in these areas.
The first question is in what department the research should take place? The answer is both education (learning science, cognitive science, etc.) and computer science. The most successful teams will be those pulling together people from both departments. Researchers need deep expertise in computer science as well as a robust understanding of the types of questions and methods used in education. If computer science departments fail to take a leadership role, computer science instruction will continue to suffer from a gap in content, methods, and tools.

We can look in history for two examples: engineering education and computational biology. Preparation of physics, math and science K-12 education largely happens in education departments or schools because these core subjects are required in K-12. Engineering K-12 education, on the other hand, has found a place in the college of engineering as well as education. Research on college-level instruction occurs in cognate departments. In all solutions, the cognate field is a large part of the research. Computational biology is another example. Initially, both biology and computer science departments were reluctant to hire in this area. CS departments felt biology was an application of current algorithms. The few departments who saw the promise of computational biology have made transformative discoveries in mapping the human genome and driving new computer science areas such as data mining. The same will hold true for computing education research. Computer scientists need to lead not only to make advances in computing education, but also to find the problems that will drive computer science research.

One might argue, then, that lecturers or teaching faculty can continue to perform computing education research. Why do we need tenure-track faculty? It is no accident that several successful transformative educational initiatives, including, Scratch, Alice, and Computational Media, have been developed by teams led by tenure-track faculty. Like any systems field, making a large impact in computing education requires time and graduate students. Lecturers and teaching faculty with high loads and few graduate students are excellent at trying new teaching techniques and reporting the effects on the grades. It is substantially harder to ask the deeper questions, which require focus groups, interviews, and detailed analytics, while juggling high teaching loads and being barred from serving as a student’s research advisor. Tenure-track positions are necessary to give faculty the time to lead research groups, mentor graduate students, and provide jobs for excellent candidates.

**CER / CS research collaborations**

One of the exciting benefits of hiring CER researchers lies in the potential collaborations with existing computer science faculty. Computer scientists are in the perfect position to partner with CER researchers to accelerate the research process through automation, create new environments for learning, and create new curricula. Performing research in the space of Pasteur’s Quadrant\(^1\) can change the

---

\(^1\) Research that is between pure basic and pure applied research; a quest for fundamental understanding that cares about the eventual societal use.
world. Like computational biology, what begins as an application of the latest in computer science can grow into problems that drive computer science research.

*Driving computer science and education research*

Computer scientists can harness the power of automation, as long as they do not lose sight of the root questions: How do students learn, and how should we teach? The relatively new phenomenon of massive data collection (MOOCs, Khan Academy, etc.) has distracted computer scientists with a new shiny set of data. Using project snapshots, machine learning can model the paths students take in solving their first programming assignment[3]. Early struggles on this first assignment were correlated to future performance in the class. Tantalizing, but correlations are meaningless without understanding why they exist.

This represents a research methods gap. Small-scale research, with focus groups, direct observation, etc., can answer *why* students follow certain development paths, identifying the mental models involved in order to inform curricular content and student feedback. Large-scale research can tell *how often* such problems occur, as well as identifying when they occur in real time. The real power, however, is merging the two approaches; machine learning can identify anomalous actions in real time and send GSRs over to talk to those students in order to discover the conceptual cause of their development path.

*Transforming Society*

Some may say that our existing system produced successful computer scientists, so why should we spend this effort pushing the efforts to K-12? Our current flawed system has produced many fewer successful computer scientists than it could have. As Warren Buffet stated generously, one of the reasons for his great success was that he was competing with only half of the population (Sheryl Sandberg’s Lean In). Research has shown that more diverse employees create better products, and companies want to compete. The work of Jane Margolis in *Unlocking the Clubhouse*[2] has inspired a generation of researchers to study how students with different backgrounds (gender, ethnic, socioeconomic) experience traditional computer science instruction. How should teaching methods, curriculum, IDEs, etc. look when designed for those neither confident in their abilities nor already viewing themselves as computer scientists? This mentality created Exploring Computer Science (ECS) and our interdisciplinary summer camp, Animal Tlatoque[1], combining Mayan culture, animal conservation, art, and computer science. We specifically targeted female and Latina/os students who were not already interested in computer science. Our results after three summers were that the targeting through themes worked (95% female/minorities, 50% not interested in computer science), and that the Scratch-based camp resulted in increased interest (especially among those not already interested) and high self-efficacy. This challenge – to reach students who are not proactively seeking computer science – continues in our development of KELP CS for 4th-6th grades (perhaps the last time when computer science can be integrated into the normal classroom for all students).
Another set of fundamental questions involves the relationship of computational thinking to the rest of education. Some researchers have claimed that coding skills improve problem-solving skills in other areas, but there is no research to back up the claim. Does learning programming, debugging, or project design help in other areas of development, such as logical thinking, problem solving, cause and effect, or empathy? Is time devoted to computer science instruction taking away from the fundamentals, is it providing an alternate motivation for students to build on the same skills, or is it providing a new set of skills that are necessary for innovators of the next century? These are fundamental questions that need to be answered, and computer scientists have critical expertise to answer them, as well as a vested interest in the results.

Departmental Benefits

What will investing in computing education research bring the department? CER has promise with respect to teaching, funding, and department visibility.

Department teaching: CER researchers, whether they perform research in K-12 or undergraduate education, often know about techniques to improve undergraduate teaching. Attending SIGCSE and ICER exposes researchers to the latest instructional and curricular techniques for undergraduate education, such as peer instruction[4] and pair programming[5].

Funding: The interdisciplinary nature of computing education and diversity of the type of research (Kindergarten through college and theory through deployments) provides a plethora of funding opportunities in two directorates of the National Science Foundation (NSF): Educational and Human Resources (EHR) and Computer & Information Science & Engineering (CISE). With limited funding per program, CISE core calls clustered in November, and strict PI submission limits, such diverse offerings can be beneficial to a faculty member with a broad research portfolio.

External Visibility: Due to the repeated calls for more computer scientists, both at college and K-12 levels, teams that combine research and deployment can bring schools substantial visibility. In the past several years, computer science departments have made headlines for MOOCs, increasing female representation, and large-scale deployments in K-12 for the purposes of social equity, all areas in the CER domain.

Conclusions:

The time is right to provide resources for computing education research. Computer science departments have sat on the sidelines for too long, reacting rather than leading. These efforts greatly affect our current and future students – the future of our field. Seize the opportunity now to make a mark on the future.


