Conversations with a Prominent Propagator: Beth Simon

David P. Bunde
Zack Butler
Christopher L. Hovey
Cynthia Taylor

Spreading the meaningful usage of transformative teaching practices in Computer Science (CS) undergraduate education requires intentional planning and sustained effort. This article presents another installment in the series of interviews with *prominent propagators*: developers who have successfully disseminated their pedagogical or curricular projects within the CS education community [1-3]. The goal is to capture experiences and insights that others can use to propagate their own educational innovations.

In this article, we interview Beth Simon, Teaching Professor in the Department of Education Studies at the University of California San Diego, who is best known for her work on Peer Instruction in Computer Science [4, 6–8]. Beth began her career with research in computer architecture and gradually shifted her focus to computer science education, officially transitioning to the Education Studies Department in 2016. She now teaches courses for people who want to teach K-12 Computer Science.

Below are highlights of the interview, which ran approximately an hour. The transcript has been edited for clarity and style.

Q: How did you get into propagation?

BS: All of my goals have been about improving the student experience—whether that's learning, or social, or a combination of both—usually in the Computer Science classroom at the university.

I would say that my first experience with propagation was three years out from getting my PhD. Steve Wolfman told me about his PhD research, which was using a tablet PC. For computer architecture, you have these diagrams where you have to show the circuit patterns. And I was like, I've been printing out overheads and carrying these markers and I have blue and purple hands. A tablet is perfect for teaching computer architecture. That got me started, but I had zero evidence or data on anything. I just thought other people who teach computer architecture should know about this thing. They were having a big conference here in San Diego and the first ever architecture education workshop was the day before. I submitted a little three-page usage paper and then demoed how I used the tablet. For us, it was game changing. It's like, “Oh, I don't have to carry all those things anymore.” That was Classroom Presenter [5] and we morphed it into a student engagement system, but it didn't go very far. I learned my first basic thing about propagation: if it relies heavily on a particular technology, you're gonna be screwed.

Q: What else did you learn from propagating Classroom Presenter?

BS: Classroom Presenter made the base case easy to adopt, which was “stand up and lecture.” When we asked people to add activities where the students would be able to submit things from their devices, that was the part that fell down because we were thinking about it from the perspective of software engineers. We built and used the system, but we didn't really address the pedagogy behind it. We spoke to it from a technical point of view rather than a “You need to change what happens in your classroom” point of view, even though that was what we were using it for.

2153-2184/2022/December – ArticleNumber
https://doi.org/10.1145/3571090

Q: How did that impact your future propagation work?

BS: I began to understand the importance of generalization and how many things will challenge people in picking something up. We had a grant from the NSF to try to propagate Classroom Presenter. We took faculty applications and said, "You're coming in as a cohort; we're gonna do three days in the summer, then you do these activities in your class and report back out." Nobody ever did anything. Two people who were accepted didn't even show up—no communication, nothing. That was a very rude awakening for me that professors are human too.

What I think has worked better and which the research would say has worked better, is more homegrown propagation. In three days, you can't convince any faculty member of anything, especially if it requires them to do work changing how they teach.

Q: What does "homegrown propagation" mean to you?

BS: It's better to convince people in-house: the people that you're hanging in the hallway with. Faculty, even at large places like UCSD, have students who took another person's class who go on to take their class and say, "Your lectures would be a lot better if you did this thing." That usually doesn't go anywhere in one quarter, but it builds up over time.

But what really was the propagation leverager was the fact that we had a program to let graduate students teach in the summer session. They had the time and the incentive to create a collection of Peer Instruction materials, which started making its way into their advisors' classes. That became a game changer. I think that was by far the most important thing in terms of propagation: all those people graduating and taking Peer Instruction to a whole range of institutions was really what made things happen. How many places are there with a large number of graduate students that you can influence over multiple years of time and have time to practice with? I'm not sure how many that is, but certainly in this case, I think it was the magic sauce.

Q: Is there a way to scale up beyond one institution?

BS: Somebody convinced me to do the New Faculty Workshop based on what they do in physics. [Before COVID,] it was really starting to swing in positive ways in that we were starting to have people coming who were way more interested. They were coming in specifically because they were going to go to R1s, but they were like, "I want to learn Peer Instruction."

But it's one of those things that it takes you forever to get feedback, probably 10 years, on what people are doing. Tracking usage, tracking adoption, let alone fidelity of adoption, is something I'm not interested in doing.

Q: How do existing faculty learn about new teaching methods?

BS: The classroom before the pandemic was the most unobserved, sacrosanct place. It was just not acceptable for us to go into other people's classes. At most, we complain to each other at the printer. There's now, in the CSE department, a "Talking Teaching" listserv that Joe Politz runs and people are asking questions about the classroom, but that was not something you did before. It was like, "Figure it

out yourself or ask your TA to figure out how to do something." I'm very glad that at some schools that's changing a lot. It's more like we're in this together. Teaching is really hard. Optimizing human learning is really, really hard. None of us can figure out the magic sauce by ourselves.

Q: What did you see as your role in propagating new CS education techniques?

BS: It was to be an individual who hopefully had the big-enough classes. When I started doing research with people, the majority of my colleagues were at small private institutions, which is where I did my first three years. So, I understood the challenges they had of not having enough numbers and not being able to have two sections where you could have a control and all that sort of thing.

So I used the luckiness of having huge UCSD classes to be the person who could demonstrate at scale whatever impacts we thought would be useful—whether it was exam scores or student attitude, all that sort of stuff. I can get large numbers. And that was really what I thought would do it. I thought I'd write those research papers and then those numbers would convince people.

Q: How did things change when you moved to the Education department?

BS: Before I moved, I had about 10 years worth of experience through various NSF grants of working with local high school teachers who wanted to learn to teach computer science. And that got me into some of their classrooms to really see what life was like. You can't just put a clicker in somebody's hand and tell kids to discuss and have them actually discuss because two thirds of them don't wanna be there, even if it's an AP class. It's a radically different space.

I had this one teacher I worked with who came up and sat in my lecture hall for an entire term. He saw for a solid eight weeks exactly what Peer Instruction looked like in my classroom. There is no better description of what it looks like for real. And then I go see it in his classroom and it falls apart. It was a combination of his students along with the fact that he actually doesn't have subject matter expertise, let alone pedagogical content knowledge about why I developed the questions.

I realized when I started to create my classes for K-12 teachers that I needed to be very cognizant that our teachers may not have deep subject knowledge. So you need to make sure that the things you're recommending for the time in the classroom are scaffolded in a way that the teacher can do what most of us would do if we were in a lab setting, which is use the computers to help the students learn. Use the computer as a co-teacher; you just do the guiding.

The other thing that I realized is that my experiences about what worked in college didn't [work in K-12]. You say, "This is what I do. This is why I do it. This is what I think the benefits are for students having practice talking to each other, understanding how computer scientists think." Then you ask them, "What is it that you do in the classroom that you think we could modify to make this happen?" Because they have things that they've been doing for years and years, and they have certain things that work best for their students and for them. If you happen to put Peer Instruction in front of them, and they go, "This is awesome", I can let them go. But don't in any way, shape, or form be like, "You should spend a huge

amount of time and effort and money to make some change based on what works for me in the university setting."

Q: How have you adapted university teaching techniques for use in K-12?

BS: In the last year and a half, I've benefited from working with experts on two projects for English language learners. Holy cow, I've learned so much stuff, it's great! There's one technique that we call coded commentary. You have kids plan by writing comments. It's an English language learning technique, we just gave it a computer science name. It's a little bit different because it's code—it's not just writing things on paper; you can actually run it and test it. So, we modify some of their techniques and give examples of what it looks like in the Computer Science classroom.

We have a teacher expert who we brought in on our first year to help teach us. She has a master's degree in English Language Learning Support, and she's taught computer science for five years. She's doing these things with us and she said, "This is what I would train teachers to do—these English language learning supports in the classroom. I did it all the time, but then suddenly I taught computer science, and I stopped doing it. I couldn't really tell you why, except that I was just so overwhelmed with the content knowledge and getting it to the kids." Then she's like, "Maybe we just need to bring that up to people, that they probably already know how to do this. 'Have you ever done something like this in a math class?' Give them examples: here's what it can look like in Computer Science."

And so I would say that the big difference in doing propagation with in-service teachers is leveraging the things that they've already been doing that they were taught when they were getting their teaching credentials. I will also say that Computer Science teachers, at least in high school, suffer from the exact same things that we do in the university, which is that they have more content than there is reasonably time to fit in.

Q: What has helped propagate techniques outside of Computer Science?

BS: Just what does Peer Instruction look like in biochemistry? There's not really much of anything we do that's actually Computer Science-specific. The biggest thing that causes change is having people whose job it is to create materials for faculty, to be collaborators with them, to co-teach with them in the classroom, or in any other way support them. Having graduate students or [education] experts who can sit in the classroom and be that person who says, "I know it's really loud. It's okay. It's supposed to be like that." That really helps with adoption.

I think it would make a huge difference if we could change the whole world so people were thinking about their teaching as a research experiment, like "What can I do now? What's going on in my classroom? Give me some feedback. What do you see that's not working?" I would benefit from having that outside expert as well, somebody who has a variety of tools in their toolbox who can be like, "Maybe try this?" Also having that separate person there to observe, because we all stink at observing our own classrooms. We can't remember what we did in the classroom at the end of the term, or the end of the period.

Q: How has the field of CS Education Research changed over the course of your career?

BS: Computer Science Education wasn't a thing that we paid attention to at all 20 years ago. I think that has maybe changed. Obviously, [it has changed] in some departments more than others, but there's not just one way to do it. It's super cool to see that in the UCSD CSE department, there are so many people who do CS Education and they're now given cred. They started matching up a senior faculty member,

whose research area is not CS education, with a Teaching Professor who specializes in CS education. Together they co-teach CS1. The Teaching Professor is the lead teacher and the professor without CS education knowledge is there to learn. Then maybe they take those techniques and use them in their other classes.

Q: What does successful propagation look like for your projects?

BS: I feel pretty comfortable about what is successful with students based on very practical things that we care about, like do students continue on? Do they do well? All that stuff. Successful in terms of propagation? As much as I would love to track and make sure that things that are being done are going okay, I think it's just really hard. I quickly realized that people could implement your strategy or your technique poorly, even if you thought you were being very clear about how it worked. We just try our best to make sure that people don't do the obviously wrong things. I always remember that when we first made "What not to do when you're doing Peer Instruction" [6], we filled up a page of small text and we could have gone on, and I was like, "Ooh, this is a problem."

Q: What advice would you give to someone trying to encourage adoption?

BS: Look for whatever the most common pain point is amongst the people that you want to reach. That could be different for different people. We found this in the New Faculty Workshop: what they were stressed about as a new faculty member was really different for people who were going to R1s versus people who were going to teaching schools. If it doesn't address their pain points, they're probably not going to adopt it. Sometimes you only have to change your "sell" or your explanation, which is exactly what we did in the new faculty workshops for those different groups. In the R1 workshop, it was "You're going to get better teaching evals and you'll be less frustrated." For the teaching schools workshop, it was "Don't you want to know how well they're learning?"

It is definitely about speaking to their pain points and showing them how your thing is going to make their life easier. I think especially if it's about being in the classroom, a lot of faculty feel uncomfortable about their time in the classroom and you can show them how to have a better experience.

References

[1] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2021. CONVERSATIONS: Conversation with a prominent propagator: Beth Quinn and Stephanie Weber, EngageCSEdu. ACM Inroads. 12, 4 (2021), 6–9.

[2] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2022. CONVERSATIONS: Conversation with a prominent propagator: Mark Guzdial. ACM Inroads. 13, 1 (Feb. 2022), 6–9.

[3] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2022. CONVERSATIONS: Conversation with a prominent propagator: Monica McGill. ACM Inroads. 13, 2 (2022), 14–18.

[4] Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., Zingaro, D. and Simon, B. 2016. A Multi-institutional Study of Peer Instruction in Introductory Computing. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (New York, NY, USA, 2016), 358–363.

[5] Simon, B., Anderson, R. and Wolfman, S. 2003. Activating Computer Architecture with Classroom Presenter. *Proceedings of the 2003 Workshop on Computer Architecture Education: Held in Conjunction with the 30th International Symposium on Computer Architecture* (New York, NY, USA, Jun. 2003), 11-es.

[6] Simon, B. and Cutts, Q. 2012. Peer Instruction: A Teaching Method to Foster Deep Understanding. *Communications of the ACM*. 55, 2 (Feb. 2012), 27–29.

DOI:https://doi.org/10.1145/2076450.2076459.
[7] Simon, B., Esper, S., Porter, L. and Cutts, Q. 2013. Student Experience in a Student-centered Peer Instruction Classroom. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (New York, NY, USA, 2013), 129–136.
[8] Simon, B., Parris, J. and Spacco, J. 2013. How we teach impacts student learning: peer instruction vs. lecture in CS0. *Proceeding of the 44th ACM technical symposium on Computer science education* (New York, NY, USA, Mar. 2013), 41–46.
[6] Things Not To Do | Peer Instruction for Computer Science. http://peerinstruction4cs.com/things-not-to-do.

David P. Bunde
Knox College
2 E. South St
Galesburg, Illinois 61401 USA
dbunde@knox.edu

Zack Butler
Rochester Institute of Technology
Rochester, NY 14623 USA
zjb@cs.rit.edu

Christopher L. Hovey
University of Colorado Boulder
1045 18th Street, UCB 315
Boulder, CO 80309
hoveyc@colorado.edu

Cynthia Taylor
Oberlin College
10 N Professor St
Oberlin OH, 44074
ctaylor@oberlin.edu