Conversations with a Prominent Propagator: Barbara Ericson

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

Increasing the meaningful usage of transformative teaching practices throughout 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. The goal is to capture experiences and insights that others can use to propagate their own educational innovations.

In this installment, we interview Barbara Ericson: winner of the 2022 SIGCSE Award for Outstanding Contribution to Computer Science Education; Assistant Professor of Information, School of Information; and Assistant Professor of Electrical Engineering and Computer Science, College of Engineering at the University of Michigan. Barbara's work focuses on embedding dynamic learning activities informed by educational psychology into the design of open-source, interactive eBooks on the Runestone Academy platform for introductory computing courses [2, 5, 6]. Barbara mines the data to close the feedback loop and improve the instructional materials, as well as to generate knowledge [4].

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

Q: How did you get started with eBooks and Parsons problems?

BE: For years I was a research scientist trying to help teachers get up to speed with teaching computing. And I found that they could not buy books except for once every seven years. I could teach them something interesting but they still had these old books and they would go back and teach the way that they'd been teaching. I thought that we need to make books free if we really want to make a dent in secondary computer science and how it's taught. I started looking at interactive eBooks platforms and happened upon Runestone Academy [10].

Over the years we've added a whole bunch of new features to Runestone, including Parsons problems [3], in which people put mixed-up code fragments in order. We know that people learn best if we can manage cognitive load to improve learning. Writing code from scratch can be overwhelming for beginners. Parsons problems give you all the correct code so you don't have to write it—you just have to put it in order. So I figured that seemed like a good, lower cognitive load kind of practice than writing code from scratch.

There were things people had researched about Parsons problems, but there were also things that we didn't know. So I said, oh, that sounds like a good PhD project, so we ported the js-parsons library [7] over to Runestone and then I added Parsons problems to one of the existing eBooks. And then we looked at the log file data and found that people were trying to solve them, which was exciting. More people tried to solve them than nearby multiple choice questions, but there were people who took a surprisingly long time to solve them and some people never solved them. That's what gave me the idea of trying to make the problems adaptive so that if you are struggling to solve them, the system can make it easier for you.

2153-2184/2023/March – ArticleNumber $15.00
Copyright held by the owner/author(s). Publication rights licensed to ACM.
https://doi.org/10.1145/3583091

Q: Why interactive textbooks?

BE: Because instructors need help and they don't want anything that costs them more time. If you can save instructors time and build in features, that'll be useful for them. I think that increases your odds of adoption.

Q: Have you been adding more features to the eBooks?

BE: One of the interesting things that's going on is Brad [Miller, founder and main developer of Runestone] is integrating PreTeXt, which is a math format for eBooks. PreTeXt has the capability to print out a PDF, which I was not all that excited about until one of my PhD students did a study with people who are neurodiverse using an ebook to learn Python [8]. He found that all of them wanted to print it out. There were several reasons. One is if they're online, they get distracted very easily because they can easily Google stuff or whatever, so they find paper less distracting. Some of them like the physicality of highlighting things and writing notes in the margin.

Q: How do you find out what's useful for people?

BE: When I first started working with high school teachers I went and talked to a bunch of them. I asked, “What do you need for Advanced Placement Computer Science?” They told me they needed more multiple choice questions because the AP Computer Science A exam was half multiple choice questions and half free response. And they could buy the Barron's Book [study guide] that would just say why the right answer was right, but not why the wrong answer was wrong. They told me that's one of the things they needed.

So, one of the first things I built for teachers was a website where I had multiple choice questions. I tried to make it so that other people could author them too, anonymously. You could say, “I want a test of 40 questions” and it would pull them in randomly from the database of questions. That was one of the first things I did that really got used—in fact, it was used so much that it would crash my server right before the exam. That was part of why I started creating what has become the CSAwesome ebook [1]. I wanted to port those questions over to the Runestone platform because it was much more scalable. It was handling the load much better than the old software that I'd been working on for years. I first just released it to teachers in Georgia, and then I said, “Hey Brad, is everything okay?” And he's like, “Yeah, we're fine.” So, I released it nationwide.

Q: How did you encourage people to submit their questions and participate in the development side of things?

BE: When I was doing teacher workshops, I would ask teachers to create questions. What I've found from working with teachers is that there are teacher leaders. I had the misconception that every teacher creates things. Turns out, no: often there are teacher leaders who create things like lesson plans or tasks or whatever, and then they share them.

Eventually, I learned that what I should really do is develop teacher leaders. I started sending teacher leaders to conferences and paying them to run workshops, to add materials to the books, and to review books for me. I found that teachers would value other teachers’ advice more than my advice. I could say something and they take it with a grain of salt. But if another teacher who actually was teaching the material said it, then they would be like, “Oh, okay”. It gave me more cred[ibility] if I had teachers spread the word.

Q: Was there a recruitment process for the teacher leaders?

BE: As I was giving workshops, I started noticing the teachers who were sharing materials, who were building materials. Then I would recruit them. I had them run summer camps for me, because one of the things I was doing was trying to figure out what got kids interested in computing. Trying to develop people who potentially would be interested is a good way to recruit, and then either pay them to go to conferences, or pay them for some of their time to develop materials, or pay them for answering questions if you're scaling.

Q: What was the biggest challenge that you encountered?

BE: One challenge is getting teachers to actually do what you're teaching them. I would teach people and they'd be all excited and then I'd go visit their classrooms in the fall and they'd be doing the same thing they'd always been doing, because a one-week workshop really isn't enough. That's also partially why I hired teachers to run summer camps; it was a way to get them more familiar with the things I wanted them to do and try it out in a low stress environment where they didn't have to give grades. I found that teachers were more willing to adopt new things if they had tried them out in the summer camps, and then those teacher leaders would create materials for other people and run workshops in their districts.

Q: What's been the most rewarding thing about doing this type of work?

BE: When people tell me that they use something I have created, they find it helpful—that is why we do this. In my class, I use a lot of active learning: lower cognitive load practice, adaptive practice. I say, “Look, you're gonna learn, you're gonna have to practice. And this is what kind of things we do,” and I'll have students tell me, “This wasn't as scary as I thought, this was pretty cool.” That is what I'm after: I want everyone to have an enjoyable experience with programming. Not the typical traditional style where we throw you in the deep end and see if you swim.

Q: How do you recruit people to use your eBooks?

BE: Try to help people, make it easy for them to change. Keep nudging them. You can't just tell people they're doing it wrong, that doesn't work. I've tried to be sneaky by creating free eBooks for people. I started meeting with other instructors and saying, “Hey, I create free and interactive eBooks, and I research them. Is anybody interested in a free, interactive ebook?”

I try to create the best materials I can. During my *Georgia Computes!* days [6], I was creating things for teachers and trying to figure out how to make it easy for them to understand. The nice thing about working with teachers is that they're very honest. They'll tell you, “This is not working for me” or “This *is* working.”

Q: So what types of things did K–12 teachers give you pushback about?

BE: Teachers, particularly secondary teachers, use pacing guides and lesson plans. They need a whole bunch more materials than what we usually create for undergrad instructors. It's a totally different situation. They work in smaller groups, much more one-on-one than we do as instructors. Learning from teachers—what works for them and what doesn't—is important. Secondary educators went to school for “how to teach,” which most CS professors or instructors did not. So there's that whole pedagogical thing about lesson plans with, “What are the learning objectives? What will students know and do?” which is not the way most undergrad instructors look at it.

Q: How is propagating to undergrad instructors different from propagating to K-12 teachers?

Many people don't realize that there's potentially better ways of teaching than just lecture with writing code from scratch. I keep trying to meet with [instructors] and get them to adopt. It wouldn’t work for me to just criticize them. That's why I try to help them with free books and doing some of the work to try to lighten the load on them. The biggest thing I'd like to get the word out about is if you want to make just a tiny little change, consider interactive eBooks because they will provide more interactivity for your students. You don't have to do a whole lot.

I'm trying to get some people to adopt some of the things I've done. My winning an award might have made an impact, but just because you won an award doesn't mean anybody's paying attention to you or listening to you. Some people in the School of Information adopted a little bit, but often it takes a bit more handholding. People are satisficing: if I'm doing a good enough job, why would I spend any more time on it?

Q: How can you encourage people to spend more time?

BE: We have this money, Renew CS, to try to improve the pass rates for women in introductory CS in Michigan. So there's a little bit of push from above, but they tend to think the solutions are things like a new intro course or hiring more female TAs. They don't tend to think the solution is to modify how we're teaching.

I think you gotta get people to take a good hard look at what's really happening. You also have to get past that “It's okay if we're failing a lot of people, because that's rigor” mindset. But we're failing a much higher percentage of women or underrepresented minorities. Is that a good thing? A lot of people are like, “sure it is.” Trying to change that attitude is tough.

So if you can put some money into it to say, Hey, we are gonna measure this. We're gonna put some pressure on you to do something about this and then try to disseminate best practices among the R1s.[1] The R1s look at each other. I mean, this is one of the things I learned in *Georgia Computes!*. One of the ways you get states to do stuff is you tell 'em some other state is doing better, especially a nearby state. I think one of the yard sticks for CS departments is other CS departments, like seeing Harvey Mudd increasing their percentage of women. But then people are like, “well, that's just a one off.” So you have to have it happen in multiple places for people to reconsider what they're doing. Then it takes time for people to change. You've gotta send them to workshops and give them the time and financial support to change their materials. I mean, most of the people who are teaching were not taught how to teach.

Q: What advice would you give somebody who's just getting started propagating their innovations?

BE: Make it free, make it easy for people to adopt. Run workshops or things to get people to know about it. Publish about it so people learn about it. But it's very funny: when I was running summer camps, I was gathering evidence about what was working. I had instructors come to my workshop on how to run summer camps, and did they care about the research and what we found was effective? No. What they cared about was what they found the most interesting.

So when mobile app development came out, they were all hot to do mobile app development, even though we had no research on it yet. So even though we're Computer Science Education people, we

[1] “R1s” refers to doctoral universities with very high research activity in the U.S.

don't always pay attention to the research either. Unfortunately it's not always the best thing that wins out. It's the thing people are excited about.

Q: For you, what does successful propagation look like?

BE: Getting more people to do it, not just me. Particularly if I get other people to take it over and own it like CSAwesome where I don't have to own that anymore. That's wonderful. Or when I was running *Georgia Computes!* and I got other teachers to run teacher workshops and stuff. And they were still using the curriculum I created. To me, that's the real test of propagation: does it live beyond me?

Q: How long does it take for a project to reach that stage?

BE: For MediaComp [7], I'm gonna guess it was about five years of us running workshops before other people started running workshops. The book that became CSAwesome, I was working on that for years before that got taken over. I'm gonna say about five to ten years typically.

Q: Can you explain a little bit about some of the financial considerations of some of these projects?

BE: Sure. Stuff takes money, even the free stuff. Runestone is free. One of the battles has been finding enough money to keep Runestone going. Free things don't necessarily mean that they don't cost anybody anything. They take time and energy, so we have worked on grants to try to keep at least the minimum stuff going on.

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

BE: For people that are willing to try things to improve their courses, start small—do something easy. Add Peer Instruction [9]. That's an easy thing to start with and we already have lots of evidence that it is very effective. So if you're willing to take baby steps, take baby steps and then learn more, go to the community, read, find out what's effective, try things. One of the things I love about SIGCSE: I've always learned something new at SIGCSE.

Q: What motivates you to work on all of these projects?

BE: I care about learning. I love programming. I think computing is fun, interesting, and exciting, and there are a lot of jobs in the field. The mid-80's was when we were at the highest percentage of women. Some of my first jobs, we had eight programmers with a token man. So, how much has the field changed from then? We went from nearly 40% women down to often under 20% at universities. I just think that's a shame and I'd also like to see more racial diversity. That's part of what drives me.

I always like that story about people throwing a starfish back in the ocean, and somebody saying, "Well, you're never gonna make much progress doing that." And you're like, "Well, it affected that one."

So yeah, stuff takes work. Nothing happens without work. You've gotta put in work and passion. This is something I'm passionate about. That's what makes it worthwhile, that you care about it and you want to see the results.

References

[1] CSAwesome: AP CSA (Java) Curriculum: *https://www.csawesome.org/*. Accessed: 2022-11-18.

[2] Ericson, B. and Haynes-Magyar, C. 2022. Adaptive Parsons Problems as Active Learning Activities During Lecture. *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (New York, NY, USA, Jul. 2022), 290–296.

[3] Ericson, B.J., Denny, P., Prather, J., Duran, R., Hellas, A., Leinonen, J., Miller, C.S., Morrison, B.B., Pearce, J.L. and Rodger, S.H. 2022. Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs. *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (New York, NY, USA, Dec. 2022), 191–234.

[4] Ericson, B.J., Maeda, H. and Dhillon, P.S. 2022. Detecting Struggling Students from Interactive Ebook Data: A Case Study Using CSAwesome. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (New York, NY, USA, Feb. 2022), 418–424.

[5] Ericson, B.J. and Miller, B.N. 2020. Runestone: A Platform for Free, On-line, and Interactive Ebooks. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (New York, NY, USA, Feb. 2020), 1012–1018.

[6] Ericson, B.J., Rogers, K., Parker, M., Morrison, B. and Guzdial, M. 2016. Identifying Design Principles for CS Teacher Ebooks through Design-Based Research. *Proceedings of the 2016 ACM Conference on International Computing Education Research* (New York, NY, USA, Aug. 2016), 191–200.

[7] Guzdial, M. 2013. Exploring hypotheses about media computation. *Proceedings of the ninth annual international ACM conference on International computing education research* (New York, NY, USA, 2013), 19–26.

[8] Haynes-Magyar, C. 2022. *On Learning How to Program via an Interactive eBook with Adaptive Parsons Problems*.

[9] Porter, L., Bailey Lee, C., Simon, B. and Zingaro, D. 2011. Peer instruction: do students really learn from peer discussion in computing? *Proceedings of the seventh international workshop on Computing education research* (New York, NY, USA, Aug. 2011), 45–52.

[10] Runestone Academy: *https://runestone.academy/*. Accessed: 2023-01-26.

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