Conversations with a Prominent Propagator: Susan H. Rodger

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

Encouraging faculty to adopt new, high-impact teaching practices, tools, and curriculum in computer science (CS) undergraduate education requires intentional planning and sustained effort. This article is the next installment in the series of interviews with prominent propagators: members of the CS education community who have successfully spread pedagogical or curricular innovations [4–6]. The goal is to capture knowledge and experiences that others can use to propagate their own teaching projects.

In this article, we interviewed Susan H. Rodger, a Professor of the Practice at Duke University. Susan has served as Chair of ACM SIGCSE and chair of the AP Computer Science Development Committee. She currently serves as Co-Chair of the Computing Research Association Committee on Widening Participation (CRA-WP). She has autism and is a powerful advocate for inclusive computing.

In this interview, we discuss her work propagating the JFLAP tool for teaching formal languages and automata [12–14, 17], the Alice programming language [1, 7, 15], and Peer-Led Team Learning (PLTL) [11].

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

Q: For those who might not be familiar with it, how would you describe JFLAP and how did it get started?

SR: JFLAP is software for experimenting with types of automata and grammars. And it actually goes further than that by letting you experiment with proofs and an application of parsing.

My first teaching position was at Rensselaer Polytech Institute as a new assistant professor. Right before I got there, they had decided that they wanted to combine Automata Theory with CS 1 and CS 2. They asked another new instructor, Ellen Walker, and me to implement it. We were naive and agreed. It was actually quite a challenge, but we were up for it.

What I found was that students were writing finite state machines on paper and turning them in, and there was always an error because it's just too tedious to do that. It was also a pain to grade. So that's where I thought, there should be a tool where they could actually program a finite state machine and run it, and then the finite state machine programs would be easier for us to grade.

So that's how building tools for automata started. The first one we built was called NPDA and was just for pushdown automata. And then eventually we figured out that the tool should be expanded to include other automata and grammars. That tool was originally called FLAP–Formal Language and Automata Package. It was a real pain for users to compile because it was written in X-windows and C++. I was getting email all the time because people had different versions and could not compile it.

2153-2184/2024/MonthOfPublication - ArticleNumber

http://dx.doi.org/10.1145/3699720

In around 1996 when Java came along, we decided to convert FLAP to Java. I decided to add the "J" on the front of the name. At the time, I thought it was stupid, but everybody was calling their new tool J-whatever. It turned out to be a really good thing, because JFLAP is not a word in any language so if you Google it, every hit is about my tool. The other cool thing was all the email I was getting from people who couldn't compile just disappeared. They could just run JFLAP. It was awesome.

Q: How did you attract your initial users?

SR: I went to SIGCSE and I published what I was doing. I also got a grant for JFLAP and included money to run a workshop at Duke. And it wasn't just automata theory, it was a visualization workshop, and there were a lot of people in the community who were also developing visualization tools. At the first ITiCSE conference, I was in a working group on an overview of visualization, its use and design [3]. I think that my workshop and that ITiCSE working group, both in 1996, helped build the community of visualization tools and educators. Then there was a series of ITiCSE working groups on visualization tools, and we also did workshops at the SIGCSE technical symposium.

We made JFLAP available for free. If you wanted to modify the software, you could do that. People did that and published papers on what they had done.

Q: When people make additions or enhancements, do those ever come back into your code base?

SR: The development of JFLAP was pretty much just me and my students. I've had 48 students who worked on it, mostly undergrads. Over time we would try new things, sometimes as a new tool. Then if we liked it, we'd add it into JFLAP. If somebody else had done something, we might also add it into JFLAP.

Q: How has collaborating with students affected your work?

SR: Students are great. Just get them involved. One of my undergrads who wrote JFLAP is the co-author on the JFLAP book [17]. I asked him, "Do you want to write half the chapters?" I thought I'd get the book out faster. He went on to get a PhD at Cornell, and now he's at Microsoft. That might be an unusual situation, but if you find the right students, it's a great experience for them.

Q: How did you get involved with Alice and what was your piece of that project?

SR: I was at an AP reading a long time ago where we were grading the AP CS exams. They usually have an outside speaker come to make the week more interesting, and Randy Pausch gave a talk about Alice — that's the first time I had heard about it. It was his first version of Alice where you actually typed code. It wasn't drag and drop at the time. I was creating a class on animation and different tools, and I ended up using Alice as one of the tools.

I wrote a paper about this class and gave a talk at SIGCSE in 2002 [16]. Wanda Dann and Steve Cooper came up to me afterwards because they were interested in using Alice in CS0. Randy Pausch was using Alice in a different way: in an interdisciplinary class for people to create animations, not really for beginning programming. Wanda and Steve saw that Alice would be cool for beginning programming, but they would need Randy to make modifications to it. They saw that I was interested in using Alice with beginners, so I became an adopter and taught a full CS 0 class using Alice with a new drag-and-drop interface.

Wanda, Steve and I started running Alice workshops with both high school and college instructors. I ran over 30 Alice workshops at Duke and four Alice symposiums. We got NSF funding, so we ran workshops in North Carolina, South Carolina, Mississippi, California, and Nebraska over the years. When our funding ran out, Steve Cooper and I got together and created an online Coursera course on Alice. The timing was great because we finished it in January 2020, right before the pandemic started.

One issue we had is that middle school is really a nice target age for Alice, but middle school students start at age 12 and you have to be 13 to take a Coursera course. Coursera wouldn't budge.

Q: What other challenges have you faced with distributing Alice online?

SR: Randy Pausch died in 2008. Carnegie Mellon really liked what he was doing and they have continued to have someone in charge of the Alice project. But what they haven't done is make it an online app. It's not hard to download, but that's an obstacle for some people. With Scratch, you can share your program on a webpage where people can actually play the game. Students want to share what they build. In Alice, after you've got your world finished, you can click a button and make a video and then you can put that video on YouTube if you want. But if you want to share an interactive Alice program, that is harder.

Q: Another educational innovation you've been active in is Peer-Led Team Learning. What has your involvement in that been?

SR: Peer Led Team Learning, or PLTL, is where undergraduate peer leaders lead the same small group of students each week in problem solving [10]. The problems relate to the topics they're learning in a class. PLTL had been used in chemistry for a long time already, and Susan Horwitz from University of Wisconsin had the idea that we should be using it in CS education. The two of us got together and we got eight universities on board. We just talked to people we knew and people we'd met at SIGCSE. We wrote a grant and got it funded. It was an effort to try to get more people to do CS because not many students were doing it at the time. That's unbelievable now.

We had workshops for instructors to come and learn PLTL and then an annual meeting. We brought students with us. The first meeting Pratibha Varma Nelson, who was a leader in PLTL in chemistry, spoke and taught us what PLTL was all about. Then we had to adapt it to computer science. Most of us were teaching either a CS 0 or a CS 1 course, but we were using PLTL in slightly different ways. For some classes, the PLTL component was mandatory, for others it was optional.

At Duke, PLTL was an extra course targeted toward women, rural students, and other underrepresented students. We got them to meet once a week to do this additional problem-solving. The undergrad leaders would develop the material with guidance from us. The PLTL students would meet with the same group of students every week in addition to the course they're taking and solve problems in groups. That seemed to work well so we published about it at the SIGCSE TS [11]. We ran a two-day CS in PLTL workshop at Duke, and we had 75 people come.

Then, Susan Horwitz wrote an NCWIT resource document on how people could do PLTL [2]. There were other people who started using it. People would ask us to come to their institution to do a half-day or one-day workshop for them to start using PLTL with CS. We also had a website where people could donate examples of their materials.

Q: What did you retain from the Chemistry version of PLTL?

SR: It's a similar idea in that it's separate from the class. Peer leaders are other students who've already taken the class, who are going to be a role model for students in your class. It's a small group that meets every week, so they bond together while learning the material. Usually, peer leaders are also TAs for the course, so students will be comfortable going to them with questions. The people who did PLTL in chemistry have books they wrote about it, and we would give adopters copies of those books, which had ideas on how you can create good materials.

Q: How did you get students to take the PLTL course?

SR: At Duke, I recruited incoming first-year women and other students who had high math scores (because that's all we had) by sending invitations to them to be part of our PLTL program. One thing we found was that men took our program because they already knew they liked computer science, so they wanted to do as much CS as they could. Most of the women took it because they got an invitation. And so we found that to get women to do CS, they just needed a little push.

Q: How do you get people to use your innovations?

SR: Workshops are the way to get adopters to use it, whether you have the in-person workshops that people travel to in the summer, virtual workshops, or the three-hour workshops at SIGCSE TS. At short workshops, they can try something to see if they want to do more. And then you have them come to an in-person workshop so that they'll feel comfortable using it or adapting it.

In Alice, we had a lot of middle school teacher adopters from many disciplines. So, for that we ran two-week workshops because these are teachers who had never used computing. The first week we taught them how to program. The second week we worked with them on developing a lesson plan, so they had something to take back to use with students. But you could still see the teachers were very nervous. I always hired a lot of undergrads to help me. The teachers loved the undergrads. The undergrads would help teach the workshop and help the teachers develop these lesson plans.

Now, the teachers are still going to want more. CMU had this Alice teacher mailing list, and Don Slater was fantastic: anybody who sent him a question, he would answer within a day. If you want people to use something, you need to respond to them when they have problems or they're going to eventually drop it. We also encouraged participants to bring another teacher from their school to the workshop so that when they went back, they would have somebody else at their school to help them. So, I think it's about support: make sure they have support in some way, and community, and a place to get help when they need the help.

Q: How do you structure a workshop?

SR: You don't want to just sit there and lecture the whole time. You need to have activities where they're trying things. In the Alice workshops, they were writing a lot of programs and they had a lot of support. The nice thing about the in-person workshops is that you're immersed with others for several days. You're together. You're talking about things. Try to do a few fun things too. If you're gonna have a keynote, make sure you get a good speaker who's gonna be really interesting. That makes the workshop a whole lot better. Create ways for people to discuss things so everybody's involved.

I would always take a few teachers out to dinner and try to get feedback from them. "Tell me, what can we do better?" That kind of thing. Definitely get feedback after the workshop. Maybe require that they fill out a feedback form before they get paid.

Get some students involved too. In the Peer Led Team Learning workshop, it wasn't just the professors coming to our annual meeting—some of their peer leaders also came for the training. For JFLAP, I would have students who were working on creating the JFLAP software be at the workshop to help people. With Alice, the students each got to present on the material they developed. I wanted to make sure their presentations were good, so I went over them with the students beforehand.

Q: What advice do you have for other people who are looking to propagate their innovations?

SR: There are working groups now at multiple conferences, CompEd and ITiCSE. Working groups are a good way to meet people, especially internationally, and collaborate. I've recently done two working groups on Parsons problems [8, 9] and we've run studies together at our universities. It's a bigger time commitment, but you do good work, you get a paper out of it, and you meet other people. I've collaborated with people from the visualization workshops for 30 years. Meet people you might collaborate with and build community is the biggest thing I could say.

If you do a panel at SIGCSE TS, you don't want to be all from the same university. Sometimes you don't want to be exactly the same topic either. You have to find a mix of people. You have to reach out to other people who might give a different perspective or have a slightly different tool. For example, the automata theory sessions at SIGCSE TS are usually low attended. Everybody wants to go to the CS 1 sessions. But if I'm doing a tool and somebody else has a visualization tool on CS 2, then they're going to come hear about my tool as well. If you have more of a variety of things, it's going to bring more people.

Q: Did seeing what other people were doing ever influence your development or your strategy?

SR: Well for me, because I built tools, it was interesting to see they have a button for something on their tool, maybe we should have a button like that on our tool, even though it's a completely different tool. So I think you learn by seeing what other people are doing. Even in other disciplines. If we hadn't looked at chemistry, we would never have started PLTL in CS. I think that's why they try to get keynotes from people in other disciplines at SIGCSE TS. It is very valuable to look at what other people are doing.

Q: What advice would you give for people looking to get funding for propagation?

SR: First, try to do some of the work so that you have something to show. That's important. For the first grant that I applied for in my department, there were a bunch of people who wanted to get together and write a grant, so it was for lots of different things and JFLAP was just one of them. We wrote a big grant. The problem with big grants is that they don't fund many of them, and we didn't get funded. I decided to just break away and do a really small grant for JFLAP, and that got funded. When you're asking for less money, it's more likely you're going to get funded.

The other thing a lot of people don't know is that if you're writing your first grant, or any NSF grant, you can always ask to see other grants that are funded. I looked around to see what was similar to what I was trying to do. I asked those people to send me their 15-page writeup, so I could see what an award-winning grant was. That was very helpful.

Another thing I like to do with grants is add in undergrad researchers. You can ask for an REU later, but why not just ask for them now so that you have that built into the grant? And I like to add a lot of workshops.

Then to build, you need to have faculty adopters. I think that's important. They're going to give you more money if you are disseminating and you have other people using it. For Alice, we added people and we had a grant where we were going to be expanding our programs in three states. So we got a much larger grant by doing that.

Q: What does a successful project look like for you?

SR: It's successful if you get faculty adopters, and people are using your tool, and they're enjoying using it. I still get emails from students who say "I just found this JFLAP tool online. I wish my instructor would've used it." I get these emails all the time.

Citations:

[1] Adventures in Alice Programming: *https://www2.cs.duke.edu/csed/alice/aliceInSchools/*. Accessed: 2024-06-11.

[2] Barker, L. and Cohoon, J.M. 2008. Peer-Led Team Learning (Case study 2): Retaining Women through Collaborative Learning. *National Center for Women & Information Technology: Promising Practices*. (2008), 3.

[3] Bergin, J., Brodie, K., Patiño-Martínez, M., McNally, M., Naps, T., Rodger, S., Wilson, J., Goldweber, M., Khuri, S. and Jiménez-Peris, R. 1996. An overview of visualization: its use and design: report of the working group in visualization. *Proceedings of the 1st conference on Integrating technology into computer science education* (New York, NY, USA, Jan. 1996), 192–200.

[4] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2023. CONVERSATIONS: Conversation with a Prominent Propagator: Carl Haynes-Magyar. *ACM Inroads*. 14, 4 (2023), 12–16. DOI:https://doi.org/10.1145/3616016.

[5] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2023. CONVERSATIONS: Conversation with a Prominent Propagator: Frank Vahid. *ACM Inroads*. 14, 2 (2023), 14–17. DOI:https://doi.org/10.1145/3594875.

[6] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2024. CONVERSATIONS: Conversation with a Prominent Propagator: Joanna Goode. *ACM Inroads*. 15, 2 (2024), 18–21. DOI:https://doi.org/10.1145/3631714.

[7] Cooper, S., Rodger, S.H., Schep, M., Stalvey, R.H. and Dann, W. 2015. Growing a K-12 Community of Practice. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, Feb. 2015), 290–295.

[8] 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.

[9] Ericson, B.J., Pearce, J.L., Rodger, S.H., Csizmadia, A., Garcia, R., Gutierrez, F.J., Liaskos, K., Padiyath, A., Scott, M.J., Smith, D.H., Warriem, J.M. and Zavaleta Bernuy, A. 2023. Multi-Institutional Multi-National Studies of Parsons Problems. *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education* (New York, NY, USA, Dec. 2023), 57–107.

[10] Gafney, L. and Varma-Nelson, P. 2008. *Peer-Led Team Learning: Evaluation, Dissemination, and Institutionalization of a College Level Initiative*. Springer Netherlands.

[11] Horwitz, S., Rodger, S.H., Biggers, M., Binkley, D., Frantz, C.K., Gundermann, D., Hambrusch, S., Huss-Lederman, S., Munson, E., Ryder, B. and Sweat, M. 2009. Using peer-led team learning to increase participation and success of under-represented groups in introductory computer science. *Proceedings of the 40th ACM technical symposium on Computer science education* (New York, NY, USA, Mar. 2009), 163–167.

[12] JFLAP: *https://www.jflap.org/*. Accessed: 2024-06-11.
[13] Linz, P. and Rodger, S.H. 2023. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Learning.
[14] Mohammed, M., Shaffer, C.A. and Rodger, S.H. 2021. Teaching Formal Languages with Visualizations and Auto-Graded Exercises. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (New York, NY, USA, Mar. 2021), 569–575.
[15] Rodger, S., Dalis, M., Gadwal, C., Hayes, J., Li, P., Wolfe, F., Zhang, W. and Liang, L. 2012. Integrating computing into middle school disciplines through projects. *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (New York, NY, USA, Feb. 2012), 421–426.
[16] Rodger, S.H. 2002. Introducing computer science through animation and virtual worlds. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education* (New York, NY, USA, Feb. 2002), 186–190.
[17] Rodger, S.H. and Finley, T.W. 2006. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett Learning.

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