Conversations with a Prominent Propagator: Carl Haynes-Magyar

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 [2–4]. The goal is to capture knowledge and experiences that others can use to propagate their own teaching projects.

In this article, we interviewed Carl Haynes-Magyar, Presidential Postdoctoral Fellow at Carnegie Mellon University's School of Computer Science in the Human-Computer Interaction Institute. Carl is the creator of the Codespec [5], online programming tool, which is designed for self-guided programming practice in a variety of modalities, including scaffolded programming tasks such as block-based languages and Parsons Problems, as well as free-form coding. Codespec is built on IDEAS+: inclusion, diversity, equity, accessibility, sexual orientation and gender identity awareness, and justice [15]. Codespec scaffolds learners' programming skills development through a range of research-based problem types and learning support features. Carl also conducts research on tools and processes that support learners' development of self-regulated learning skills [10–13].

Below are highlights of the interview, which ran approximately an hour and a half. The transcript has been edited for clarity and style, and our interviewee has generously provided citations to relevant work.

Q: Designing for a diversity of users and creating a sense of belonging seems to be central to Codespec. Can you please tell us a little about designing for a sense of belonging and how that expresses itself in your work?

CHM: Coincidentally enough, I was just talking to some trans and non-binary (TNB) computing education researchers about a workshop they held that investigated the intersectional identities of TNB learners as they flow through the ecosystem of computing [19]. LGTBQIA2S+[1] learners are more likely to think about leaving computing because of a low sense of belonging [21]. And yesterday, I talked to a group who was awarded an NSF grant and were interviewing different stakeholders about how to make coding more accessible to people with physical disabilities [6]. It's important to me to design Codespec in a way that's inclusive. Designing for a sense of belonging in part means having conversations with as many people as possible who are working to increase IDEAS+. This is especially important given "Don't Say Gay" laws, anti-trans and anti-critical race theory bills [in the United States], and the rise of large language models like ChatGPT.

[1] This acronym broadly refers to diverse gender-, sex-, and sexuality-based identities including Lesbian, Gay, Transgender, Bisexual, Queer, Questioning, Intersex, Asexual, Aromantic, Two-Spirit, Gender Fluid, etc.

2153-2184/2023/MonthOfPublication - ArticleNumber $15.00
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
http://dx.doi.org/10.1145/3616016

I'm also a part of AccessComputing and they've had several researchers, such as Neil Brown, address the inaccessibility of block-based programming [1]. Accessibility is always on my mind. And because of the conversation I had yesterday, I started thinking more about how Codespec can be designed for people with physical disabilities. The great thing is that I threw IDEAS+ in there from the beginning and I have my whole career to collaborate with other researchers to address the needs of different learners in computing education.

One of the things that's different about Codespec is that I designed the solution-side of the problem space area to make use of a grid because it is something that we can look at and measure more precisely. It was inspired by a Medium article, "4 Major Patterns for Accessible Drag and Drop" [9] and my background in technical architectural drawings. Blind and low-vision (BLV) programmers can navigate environments that use a grid-coding paradigm more efficiently than some go-to text editors [7]. Also, most block-based programming environments are not screen reader accessible [17]. But programmers *can* use their screen readers to interact with Codespec.

I'm continuously working to include other research findings into the design of Codespec to make the problem space area more accessible. I've spent a lot of time on cognitive accessibility to meet the needs of neurodiverse programmers—not all disabilities are visible. Stack Overflow's recent Developer Survey included demographic information about programmers with disabilities, and the percentage of people who identify as neurodiverse has increased from 20% to 22% [20]. Most of the research on the accessibility of block-based programming has focused on programmers with visual or motor impairments [17]. There is relatively little research in computing education on learners with cognitive disabilities [16], so I'm glad we have data on neurodiverse programmers to make their needs more visible. Part of designing for a sense of belonging is reading and staying abreast of statistics.

Q: How are you envisioning scaling up and getting new users for Codespec?

CHM: I've started by conducting research and recruiting novice programmers taking computer science courses at Colorado State University, Ramapo College of New Jersey, Syracuse University, the University of Illinois Urbana-Champaign, the University of Michigan, the University of Washington, and Carnegie Mellon University. I almost recruited students from a university in Chile but didn't because I couldn't find a Spanish-speaking research assistant.

Little tangent—another thing Codespec will offer is the ability to switch between programming languages *and* spoken languages for each problem. For example, a programming problem in Python and Java in both English and Spanish. We know that non-English speaking programmers may experience barriers when reading and writing code [8]. I was talking to somebody not too long ago about the beauty of other languages like Mandarin—how the differences in tone can mean differences in meaning. There are things we lose out on because the majority of programming languages are in English—it's something to think about.

And then I've also connected with Ms. Chevonne Hall, CEO of the Baltimore Leadership School for Young Women (BLSYW), to use Codespec in the AP Computer Science class this fall. They currently prepare students for the AP CSP exam and are exploring how they can prepare students for the AP CSA exam, which is in Java, but several computing education instructors think the exam should be switched to cover Python. To be better equipped for other careers, I've proposed using Codespec to prepare the students to earn a certificate through the Python Institute. I plan to meet with their computer science teacher to

discuss how we can support learning how to program in both Java and Python. Since Codespec supports both languages, a student can solve the same problem either way.

I also hope to connect with computer science teachers in Pittsburgh, and get instructors at Carnegie Mellon University and the University of Michigan to use Codespec in one of their introductory programming courses.

That's how I've been thinking about scaling up—through research with novice programmers taking introductory programming courses. Many of the participants go on to use Codespec afterwards. The other way I've been thinking about scaling up is by increasing the problems in the problem bank and adding an eBook for programmers interested in self-directed learning.

Q: How do you approach people who you want to collaborate with?

CHM: Networking and collaborating is easier when you attend the same conferences as potential collaborators. This year I attended the ACM Special Interest Group on Computer Science Education (SIGCSE TS) again and met several people. I reached out to one of the authors of "Computing in Support of Disciplinary Learning." I knew there was an opportunity for synergy because I'd just been awarded a SIGCSE Special Projects Grant to create AI-generated discipline-specific programming problems for Codespec. I thought I could geek out about the different dimensions of relevance: cognitive, affective, cultural, topical/disciplinary. I reached out to meet via Zoom and it was great because he geeked out as well about the possibility of creating programming problems for his students that are relevant to them. My sales experience, I think more than anything, has taught me that it's so much easier when you genuinely want to solve the same problem as the other person.

Before I meet with a potential collaborator, I think through my ideas and do my homework—get to know the person. I try to lay out my vision based on my research agenda and make sure they understand why I want to collaborate with them. I'm currently working on the IRB[2] and have just met with a colleague to strategize about how to recruit teachers from the Computer Science Teachers Association (CSTA) and different mailing lists such as SIGCSE-members, Queer In AI, and Black in AI. I'm looking forward to what we end up doing, and how the teachers end up evaluating the programming problems generated by large-language models.

After SIGCSE TS, I also met with a Ph.D. student. Again, the collaboration was easy to spark because of his work on autogenerating distractor blocks for Parsons problems.[3] When I think about collaborations, I also think about what stage of their career a researcher is in. It's good to work with both senior scholars and up-and-coming ones for exposure to different perspectives.

It's also good to have a rough timeline for things. When talking to a colleague I'd met at the annual Association for Library and Information Science Education (ALISE) conference, I made sure that he knew that my goal had a deadline that was months away. That's how I approach things a lot of the time because I know I think big—I plan way ahead. One thing that I would say is to make sure that you communicate the flexibility that you have to collaborate and what your goals are. I had asked this colleague about

[2] Institutional Review Board is a local committee at each university in the United States (that participates in federally funded research) that reviews and approves the ethical treatment of human subjects and their data.

[3] Parsons problems or Parsons programming puzzles require learners to place code blocks in the correct order and indentation [18].

translating materials into Spanish because I want to extend my research to non-English speaking novice programmers next year. People appreciate being given ample time to collaborate on projects—of course sometimes early is *too* early. I once had a professor tell me they never have conference papers finished more than two months in advance. I can't operate like that, so I reach out early and send reminders.

Q: How have user studies affected the design of Codespec?

CHM: In the past, research participants have admitted to solving Parsons problems through "trial-and-error"—repeatedly pressing the help button [14]. My research and learning analytics work with the late Stuart Karabenick, an educational psychologist from University of Michigan, changed the way I approached help-seeking when I designed Codespec's support features. I decided that instead of requiring learners to attempt to solve a Parsons problem three times before help is available, help would be available to them immediately and fade away after being used a set number of times—like lifelines. Instead of being frustrated by the inability to get help when they know they need it, learners can get help and continue problem-solving. I'd said to my co-developer, "We're missing an opportunity to capture more information about novice programmers' help-seeking strategies." So currently, Codespec allows learners to click the help button and choose what sort of help they receive. The data will inform how we can adapt help-seeking features to each learner based on their prior strategies and provide different stakeholders with learning analytics. It's healthier to encourage learners to develop good help-seeking strategies and not to learn by trial-and-error—many game shows like "Who Wants to Be a Millionaire?" take this approach and it allows us to study help-seeking in more depth. What I observed one participant do was use the help menu to choose what they need a few times, and then they didn't do so as much when solving subsequent problems. When I asked them about this, the participant expressed that they liked solving the problem themselves so they didn't want to use the help options as much.

Preliminary user studies have also led to the creation of fix-code blocks that require learners to debug code similar to lines of code in fix-code problems and the addition of block options so that learners can add custom blocks and copy code that they want to use to solve write-code problems.

An exploratory multiple-case I conducted last year with neurodiverse programmers—novice programmers with cognitive, learning, and neurological disabilities—led to four working hypotheses about how to improve the cognitive accessibility of Parsons problems for programmers with and without disabilities. For example, that programming problems which require learners to do mental arithmetic may pose challenges for learners who experience focal seizures and that Parsons problems with *paired* as opposed to jumbled distractor blocks benefit learners with attention-deficit/hyperactivity disorder (ADHD) and Tourette Syndrome. This is further evidence for the need to diversify the types of programming problems we create by considering the different dimensions of relevance and that adaptation should incorporate demographic information so that we do not create Parsons problems with distractor blocks that are far apart (jumbled) which would pose challenges for programmers who have tics that make it harder for them to click on things.

Sometimes it's hard to recruit enough people to generate findings that your colleagues deem generalizable—especially when working with special needs and underserved populations. This is something I've been trying to figure out how to do and I'm excited about figuring out how to model the individual based on the work of a colleague in psychology.

Q: What new research has Codespec enabled for you?

CHM: Codespec offers learners the option to switch between solving the same programming practice problem in several research-based ways. Programmers can practice by solving the same problem as a pseudocode Parsons problem, a Parsons problem, a faded Parsons problem (which requires learners to fill in blanks), a fix-code (debugging) problem, or a code-writing problem. Currently, I'm conducting an experiment in which participants are randomly assigned to one of four groups. In one of the experimental groups, participants are asked to solve seven write-code problems with a choice between either the pseudocode Parsons or Parsons problem as scaffolding. Hence, we're now able to investigate what novice programmers do with this agency. Codespec will also enable researchers to not only study learning rates within problem types but between them—expanding our understanding of the relationship between programming skill acquisition and problem-sequencing. This has implications for how we scaffold and personalize the learning experience for each individual.

Q: What will success look like for you with this project?

CHM: It will look like more stories about people who wanted to learn how to program and came to Codespec, felt like they belonged, and accomplished their goals. I hope they feel confident going out in the world knowing that what they've learned applies in some way, shape, or form regardless of whether or not they become professional programmers who write code from scratch. Learning how to think computationally and developing self-regulated learning skills are crucial in the 21st century.

Q: What advice would you give to other people doing the same type of work you're doing?

CHM: I would say, "Know the problem you're trying to solve." You have to know the problem space well and you have to know how to communicate its importance to scholars, practitioners, and other stakeholders to create buy-in. Do your homework. Find the people whose work is cited highly in your field and reach out to them. Know the different debates and tensions brewing. Be authentic and genuine; know yourself and know why the problem you're trying to solve is a problem for you.

References

[1] Brown, N. 2022. Blocked entry: How block-based programming is inaccessible and what could be done about it webinar | DO-IT Webinars.

[2] 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.

[3] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2023. CONVERSATIONS: Conversations with a prominent propagator: Barbara Ericson. *ACM Inroads*. 14, 1 (2023), 16–19. DOI:https://doi.org/10.1145/3583091.

[4] Bunde, D.P., Butler, Z., Hovey, C.L. and Taylor, C. 2022. CONVERSATIONS: Conversations with a prominent propagator: Beth Simon. *ACM Inroads*. 13, 4 (2022), 9–12. DOI:https://doi.org/10.1145/3571090.

[5] Codespec: *https://www.codespec.org/*. Accessed: 2023-07-14.

[6] Dawson, C. 2023. NSF Grant to Make Coding More Accessible for Persons with Physical Disabilities. *USC Viterbi School of Engineering*.

[7] Ehtesham-Ul-Haque, M., Monsur, S.M. and Billah, S.M. 2022. Grid-Coding: An Accessible, Efficient, and Structured Coding Paradigm for Blind and Low-Vision Programmers. *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, Oct. 2022), 1–21.

[8] Guo, P.J. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, Apr. 2018), 1–14.

[9] Hausler, J. 2018. 4 Major Patterns for Accessible Drag and Drop. *Medium*.

[10] Haynes, C.C. 2020. The Role of Self-Regulated Learning in the Design, Implementation, and Evaluation of Learning Analytics Dashboards. *Proceedings of the Seventh ACM Conference on Learning @ Scale* (New York, NY, USA, Aug. 2020), 297–300.

[11] Haynes, C.C. 2020. Toward Ability-Based Design for Novice Programmers with Learning (Dis)abilities. *Proceedings of the 2020 ACM Conference on International Computing Education Research* (New York, NY, USA, Aug. 2020), 336–337.

[12] Haynes, C.C. and Ericson, B.J. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, May 2021), 1–15.

[13] Haynes, C.C., Teasley, S.D., Hayley, S., Oster, M. and Whitmer, J. 2018. How am I Doing?: Student-Facing Performance Dashboards in Higher Education. *Companion Proceedings of the 8th international conference on learning analytics and knowledge* (Sydney, Australia, 2018), 274–275.

[14] Haynes-Magyar, C. and Ericson, B. 2022. The Impact of Solving Adaptive Parsons Problems with Common and Uncommon Solutions. *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research* (New York, NY, USA, Nov. 2022), 1–14.

[15] Haynes-Magyar, C.C. and Haynes-Magyar, N.J. 2022. Codespec: A Computer Programming Practice Environment. *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2* (New York, NY, USA, 2022), 32–34.

[16] Koushik, V. and Kane, S.K. 2019. "It Broadens My Mind": Empowering People with Cognitive Disabilities through Computing Education. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, May 2019), 1–12.

[17] Milne, L.R. and Ladner, R.E. 2019. Position: Accessible Block-Based Programming: Why and How. *2019 IEEE Blocks and Beyond Workshop (B&B)* (Oct. 2019), 19–22.

[18] Parsons, D. and Haden, P. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52* (AUS, Jan. 2006), 157–163.

[19] Smith, J.M., Wallace, C., Sexton, S. and Menier, A. 2023. The Trans & Non-Binary Computing Education Research Project. *ACM SIGCSE Bulletin*. 55, 1 (Feb. 2023), 6–8. DOI:https://doi.org/10.1145/3584667.3584674.

[20] Stack Overflow Developer Survey 2022: *https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022*. Accessed: 2023-06-16.

[21] Stout, J.G. and Wright, H.M. 2016. Lesbian, Gay, Bisexual, Transgender, and Queer Students' Sense of Belonging in Computing: An Intersectional Approach. *Computing in Science & Engineering*. 18, 3 (May 2016), 24–30. DOI:https://doi.org/10.1109/MCSE.2016.45.

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