Overcoming Barriers in Scaling Computing Education Research Programming Tools: A Developer's Perspective

Keith Tran
ktran24@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

John Bacher
jtbacher@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Yang Shi
yshi@ncsu.edu
Utah State University
Logan, Utah, USA

James Skripchuk
jmskripc@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Thomas Price
tprice@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

ABSTRACT

Background and Context. Research software in the Computing Education Research (CER) domain frequently encounters issues with scalability and sustained adoption, which limits its educational impact. Despite the development of numerous CER programming (CER-P) tools designed to enhance learning and instruction, many fail to see widespread use or remain relevant over time. Previous research has primarily examined the challenges educators face in adopting and reusing CER tools, with few focusing on understanding the barriers to scaling and adoption practices from the tool developers' perspective.

Objectives. To address this, we conducted semi-structured interviews with 16 tool developers within the computing education community, focusing on the challenges they encounter and the practices they employ in scaling their CER-P tools.

Method. Our study employs thematic analysis of the semi-structured interviews conducted with developers of CER-P tools.

Findings. Our analysis revealed several barriers to scaling highlighted by participants, including funding issues, maintenance burdens, and the challenge of ensuring tool interoperability for a broader user base. Despite these challenges, developers shared various practices and strategies that facilitated some degree of success in scaling their tools. These strategies include the development of teaching materials and units of curriculum, active marketing within the academic community, and the adoption of flexible design principles to facilitate easier adaptation and use by educators and students.

Implications. Our findings lay the foundation for further discussion on potential community action initiatives, such as the repository of CS tools and the community of tool developers, to allow educators to discover and integrate tools more easily in their classrooms and support tool developers by exchanging design practices to build high-quality education tools. Furthermore, our study suggests the potential benefits of exploring alternative funding models.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**.

KEYWORDS

scaling research tools, computing education tools, tool developer

ACM Reference Format:
Keith Tran, John Bacher, Yang Shi, James Skripchuk, and Thomas Price. 2024. Overcoming Barriers in Scaling Computing Education Research Programming Tools: A Developer's Perspective. In *ACM Conference on International Computing Education Research V.1 (ICER '24 Vol. 1), August 13–15, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3632620.3671113

1 INTRODUCTION

The computing education research (CER) community has developed many programming support tools over the years, such as hints, visualizations, debugging support, and teacher dashboards, to support both students and instructors, enhance the learning experience, and facilitate more effective instruction. Papers published on these tools in CER venues such as ITiCSE, SIGCSE, and ICER often present evidence that these tools can be effective, suggesting that they have the potential to improve student learning outcomes and support instruction. In this paper, we refer to these educational programming support tools that are developed by the CER community and presented in peer-reviewed CER publications, as **CER programming tools**, or **CER-P** tools for short. Although many studies point to the positive student outcome of CER-P tools, it is not clear how widely these CER-P tools are adopted in computing classrooms or what impact they have. This gap highlights the importance of exploring the barriers faced by CER-P tools developers and identifying practices to overcome these challenges. This will help to promote wider adoption and maximize the educational benefits of these innovative tools.

Promoting the use of **research** tools is important for two main reasons. First, many CER-P tools have undergone rigorous evaluation studies that show that they have a significant impact on student outcomes (e.g., [8, 11, 15, 16, 18, 32, 34–37, 44, 49, 50]). Bringing these systems into classrooms has the potential to spread these learning gains more widely. Secondly, when CER-P tools have

ICER '24 Vol. 1, August 13–15, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0475-8/24/08
https://doi.org/10.1145/3632620.3671113

more users, there are more opportunities to learn from these users through larger studies, which can advance our knowledge of student learning and improve the design and impact of the systems. Expanding the use of CER-P tools can help bridge the gap between research and practice, fostering higher quality research and better evidence-based tools supporting better learning outcomes. It is important to identify barriers and challenges that hinder the scaling and widespread adoption of these tools, as well as to explore the experiences and perspectives of tool developers.

Prior work has highlighted the importance of – and challenges of – CER tool adoption and reuse (including both tools focused on programming support and CER tools more broadly). Blanchard et al. [3] found that many CER tools are redundant, and argued against the current CER culture of recreation rather than adoption and improvement. Barker et al. [1] identified factors that *instructors* use to select CER tools for use. Hovey et al. [14] interviewed *successful* CER researchers and tool developers to identify their best practices for scaling up. However, as addressed in Section 2, three key aspects missing from the current literature are 1) an understanding of the scale of the challenge (how hard *is it* for CER-P tools to scale up?), 2) an understanding of the *barriers* that hinder the scaling up and widespread adoption of these tools, and 3) the perspectives of not only successful CER-P tool developers, but also from those who are just beginning – and perhaps struggling – to scale up.

In this study, we investigate the barriers to scaling up, practices adopted to facilitate wider adoption of CER-P tools, and propose support infrastructure for the CER community. To this end, we conducted a semi-structured interview with 16 CER-P tool developers. The interviews covered developers' objectives, the impacts of their tools, and the challenges they face in expanding their reach. We then analyzed the data using thematic analysis across the various CER-P tools to identify barriers and practices. Our findings are guided by the following research questions:

RQ1: What barriers do CER-P tool developers face in developing, maintaining, and promoting their tools?

RQ2: What practices do CER-P tool developers adopt to overcome these barriers and increase the adoption of their tools among educators and students?

RQ3: What forms of community action initiative could facilitate the broader adoption and scalability of CER-P tools?

Our research findings show systemic barriers to scaling of CER-P tools, including funding challenges, maintenance burdens, and interoperability issues, while also identifying strategies that have contributed to scaling successes. By focusing on developers' perspectives, a novel approach within this domain, our findings offer insight into overcoming these obstacles and facilitating broader adoption and scaling of CER-P tools. We argue for a significant shift within the CER community to recognize and promote the intrinsic value of educational tools based on their impact and utility, rather than on the extraordinary efforts of developers. This study emphasizes the need for systematic support and sustainable mechanisms, such as community-driven initiatives such as a CS Tool Repository and Community Network, to enhance tool engagement and value perception. Through highlighting both the challenges and the successful practices for adoption, our research shows the importance of developing sustainable support and acknowledgment systems for CER-P tools to ensure their effective utilization and greater impact in educational settings.

2 RELATED WORK

Value of CER-P Tools. The value of CER-P tools is evident within the CER community. These tools perform a range of functions, such as visualization [11, 38, 39, 39, 40], auto-graders [26, 31, 50], assessment [19, 33], and programming environment [2, 7, 20]. While each tool serves a unique purpose, the common theme among them is their facilitation of programming practice, discussed further in Section 3.1. Some studies have shown that CER-P tools can have a positive effect on student outcomes, increasing engagement and comprehension [8, 11, 15, 16, 18, 32, 34–37, 44, 49, 50]. However, not all CER-P tools have been found to be beneficial, with some showing little to no effect on educational outcomes [4, 40]. This highlights the potential for further research to support the wider adoption of CER-P tools, leading to a better understanding of diverse student users and creating better evidence-based tools.

Educator/User Perspectives. Blanchard et al. (2022) [3] conducted a comprehensive survey on *users* of existing tools in the CSEd community, focusing on the use, challenges, and barriers related to the deployment, development, and maintenance of open source and commercial software. Their study emphasizes that many CER tools are redundant, arguing against the CER culture of recreation rather than adoption and improvement. Their findings revealed that availability and discoverability are major obstacles, with educators often learning about tools through word-of-mouth or inheritance from previous courses. This perspective, although important, lacks a detailed understanding of developer challenges, highlighting the need for our work that focuses on tool developers.

Several studies have explored factors affecting the adoption of educational innovations. Barker et al. (2015) [1] investigated the factors that influence CS faculty in adopting teaching practices, but did not address the scale or barriers to tool adoption. Ni (2009) [30] examined factors influencing CS teachers' adoption of curriculum innovations, finding that teacher excitement drives adoption while organizational and social issues inhibit it. Taylor et al. (2018) [45] surveyed the existing scholarship in STEM higher education to identify factors that motivate or inhibit educators' decisions to adopt teaching innovations and argued that researchers must consider how educators will use their innovation from the inception of their design. Although these studies provide valuable insights into educator perspectives, they do not address the specific challenges faced by CER-P tool developers in creating and scaling their tools.

Developer Perspective. Prior studies on the developer perspective have been largely one-sided, focusing mainly on successful CER tool developers, such as the interviews conducted by Hovey et al. (2023) [14] with 14 CSEd researchers and developers on recruitment strategies and user base maintenance. While insights are valuable, this leaves a gap in the understanding of developers who are in the early stages or struggling to scale up. Guo (2021) [11] provided design guidelines for scalable and sustainable academic software, but did not address the specific difficulties faced by tool developers.

Although existing research has explored the views of highly successful tool developers and educators on the adoption and use of tools, it has largely overlooked the experiences and challenges

of CER-P tool developers who are just starting or struggling to scale their tools. The factors impeding the dissemination of smart learning content in CS education, as investigated by Brusilovsky et al. (2014) [5], and the challenges of aligning technology with pedagogy, as discussed by Haaranen et al. (2023) [12], further motivate the need to understand the barriers that hinder the scaling and widespread adoption of CER-P tools. There remains a greater understanding of the scale of the challenge, the barriers that hinder the scaling and widespread adoption, and the perspectives of a more diverse group of CER-P tool developers. Our work fills this void by surveying and interviewing a broad array of CER-P tool developers, aiming to better understand the barriers and potential helpful resources from the perspective of the CER-P tool developer.

3 METHODOLOGY

In this methodology section, we describe our coding review process and provide details of our participants. Initially, we conducted a survey targeting CER-P tool developers to identify potential participants and gather preliminary insights and laid the groundwork for the subsequent phase of our research. Following the survey, the goal of this work is to gain a deeper understanding of CER-P tool dissemination directly from tool developers and to conduct a thematic analysis of their responses. Thus, our approach to answering the research questions of this study involved a series of semi-structured interviews with CER-P tool developers identified in the CER community.

3.1 Population & Study Context

To identify tools that align with our inclusion criteria (detailed below) for CER-P tools, we implemented a two-step approach. Initially, we focused on curating a list of relevant CER-P tools, which are often published in computing education venues to demonstrate their educational impact. Therefore, our first step for the curation was guided by a literature review across the main CER publication venues. These venues include the ACM Special Interest Group on Computer Science Education (SIGCSE), ACM Innovation and Technology in Computer Science Education (ITiCSE), ACM International Computing Education Research (ICER), ACM Transactions on Computing Education (TOCE), and the Journal of Computer Science Education (CSE). Additionally, we also expanded our search to Educational Data Mining (EDM) and Learning Analytics & Knowledge (LAK) as some CER-P tools potentially include advanced features such as machine learning models that are published in these venues. Our review spanned publications from 2018 to 2022 (the most recent publication year at the time of our research) to focus on the most recent advancements in CER-P tools. Both research articles and "experience report" or "tools" papers were considered, since CER-P tools are published in both. An inter-rater agreement process was deemed unnecessary here for our inclusion criteria, as the initial round of review focused on broadly identifying potential candidates CER-P tools without stringent adherence to the inclusion criteria among all reviewers. Subsequently, the first author rigorously reviewed the work of the other three coauthors to ensure consistency to the inclusion criteria, with the process concluding with a final review by both the first and fifth authors, thereby maintaining a high standard of reliability without the need for formal inter-rater agreement. Additionally, our second approach involved evaluating the list of tools (both academic and industrial) identified by Blanchard et al. [3] that are adopted by computing educators for their classrooms. This process potentially included tools past five years but ensured our review was comprehensive and had a wide coverage of CER-P tools. From these two approaches, we curated a list of 146 CER-P tools that met the inclusion criteria, available online.

3.2 Inclusion Criteria

We used the following inclusion criteria to guide our curation process to find CER-P tools. The criteria included: (1) Novelty: Tools must be novel and initially created by the authors. (2) Facilitation of Practice Problems: Tools should support programming practice, supporting students during the learning process or instructors in providing practice opportunities. (3) Standalone or Integrable: The tool may either function as a standalone interface or integrate with existing programming practice environments. (4) Educational Purpose: Tools must be designed with a clear educational objective that is suitable for classroom use, or aimed at achieving educational goals for either students or instructors.

3.3 Survey Content & Distribution

To increase the participation rate, we sent out personalized tailored emails to each of the participants with reference to their specific tool and paper. Interested participants were required to complete a consent form before proceeding to the survey. The survey contained five sections focused on gaining an initial understanding of the researcher's tool, primary goals and motivation, barriers faced, and potential support needed for broader adoption. The initial survey contained 39 complete responses, where detailed discussion and findings are not in the scope of this paper. From the initial group that completed the survey, 23 participants consented to be contacted for a follow-up semi-structured interview, scheduled at their convenience. Participants were contacted at the start of Summer 2023, with one round of follow-up emails. Of these 23, 16 responded to our invitation and completed the interview. Interviews were conducted via Zoom by the first author, recorded, and analyzed for thematic content. Although our method was designed to capture a wide range of perspectives from tool developers, we acknowledge that there is a risk of participation bias due to the voluntary nature of survey completion and interview consent. We discuss the possibility for self-selection bias in Section 5.

3.4 Semi-Structured Interview

Since the goal was to understand the barriers (RQ1), practices (RQ2), and potential community-driven solutions (RQ3) associated with the deployment and scaling of CER-P tools, these interviews were designed to explore those topics in greater detail, allowing us to gather further insights while maintaining the flexibility to explore emergent themes. It is important to note that the authors' involvement in the SPLICE (Standards, Protocols, and Learning Infrastructure for Computing Education) project, which aims to develop technology and data infrastructure for CS education research, has

informed their perspective on potential community action initiatives to support CER-P tool developers. This influence is reflected in the questions asked during the interviews.

The semi-structured interview consists of four parts:

Barriers faced. The first section of the interview focused on identifying primary barriers faced by tool developers in deploying and scaling up their tools. We asked *What are the biggest challenges to deploying or scaling up the use of your tool?, what barriers did you encounter?,*

Practices employed. We asked participants two sets of question here about practices they employed: (1) *If scaled, how much of the time and effort you spent on the system went to making the system useful for people outside of your classroom,* and (2) *For each of the primary barriers mentioned, Can you elaborate on how these strategies were implemented and their effectiveness?*

Primary Resources. The third section probes participants to discuss potential community action initiatives that would help with tool dissemination. Informed by the authors' involvement in the SPLICE project, we asked (1) *Community Network of Tool Developers: What might that community look like?*, (2) *Repository: What might that repo look like?*, and (3) *Software Libraries: What are common tasks you think you and others might benefit from having libraries for (e.g. source code analysis, etc.)?* We chose to focus on these 3 specific ideas based on community interest at prior SPLICE workshops that have engaged CER-P developers, Our goal was to inform *how* these ideas might best support CER-P developers' needs. However, we make no claim that these 3 ideas are the best or only ideas for supporting CER-P developers.

Learning Experience. In the last section, we asked participants to share their learning experiences. Finally, we asked, *(1) Looking back, is there anything you would do differently in the development or implementation of your tool?* and *(2) What advice would you give to someone who is planning to develop a similar tool?*

3.5 Data Analysis

For the analysis of our interview data, we adopted an inductive thematic analysis approach. Since our focus is not on making quantifiable claims with our codes and the nature of our interview is heterogenous where each participant's interview contains rich and personalized data, we did not capture any agreement metric such as inter-rater reliability. Instead, we focused on using coding as a tool for the organization and identification of emerging themes, as discussed in Hammer and Berland [13]. Our data processing started with a joint coding session of one interview to establish a baseline. Subsequent meetings involved a team of four researchers resolving coding conflicts and agreeing on consistent tags to ensure a coherent analysis of the interview data. Our analytical process started with a collaborative coding session of a single interview transcript to establish a foundational set of codes and to calibrate our coding approach.

To support our coding and analysis efforts, we utilized Atlas.ti for qualitative data analysis. We additionally leverage memos to serve both as a tool for defining initial codes and as a record of the evolution of our code definitions over time. These memos were instrumental in capturing the rationale behind each code and in tracking the iterative refinement of our coding schema. Regular team meetings were scheduled to discuss code definitions, resolve any coding conflicts, and ensure a consistent application of codes throughout the data set. These discussions helped to achieve consensus among researchers, but also provided a forum for reflexive engagement with the data, which allowed us to iteratively refine our code definitions in light of new insights. The iterative nature of our analysis process, characterized by ongoing discussion and reflexive engagement with the data and code definitions, was informed by constructivist grounded theory methods [28]. This iterative process of code refinement involved a detailed line-by-line, code-by-code examination of each interview transcript. Through this approach, we were able to identify and elaborate on emerging themes, ensuring that our thematic analysis was both representative and grounded in the data. The absence of a quantifiable agreement metric, such as inter-rater reliability, was a deliberate methodological choice, reflecting our commitment to a constructivist paradigm where the emphasis is placed on the richness and depth of thematic insights, rather than on the quantification of coding consensus.

4 RESULTS

4.1 Context

In this section, we provide context on the CER-P tools and roadmap for the remaining sections. Table 1 provides a detailed overview and context of the CER-P tools developed by the interviewees who were part of our study. The table highlights course contexts, deployment years, estimated impact on students and instructors, tool classifications, and relevant citations. These attributes were pulled from the survey. Other results of the survey are beyond the scope of this paper.

It is important to note that the names of the tools developed by the participants are presented in this table. We recognize the potential for participants to be identifiable from this information, which was addressed and consented to in the IRB documentation. However, we chose not to link participants' names with their quotes in the analysis, as this did not further our research questions; therefore, that aspect of our data remains anonymized. The participants' numbers (e.g., P1) are not linked to the corresponding table row.

It is essential to note that our research questions can only be addressed by considering the responses from the participants in our interview. There is a strong likelihood that response bias may have affected our results, although the exact influence remains unclear. For example, it is plausible that tool developers deemed as more "successful," with higher tool adoption rates, were more inclined to participate in a survey regarding their tools. However, it is also plausible that our email invitation, which centered on barriers faced to wider adoption, could have appealed to developers encountering challenges. Further discussion of this limitation can be found in Section 5.

In the remaining three sections, we present our findings on systemic barriers to the creation, dissemination, and maintenance of CER-P tools in Section 4.2, practices developed used to address these barriers and increase adoption of their tool Section 4.3, and potential community action initiatives that would be helpful for developers Section 4.4.

System Name	CS course context	Year deployed	Estimated # of students	Estimated # of Instructors	Classification	Citation
Alloy4Fun	Formal Languages	2018	400	3	Autograder	[27]
Athene	Class Agnostic	2005	1000	10	Assessment	[48]
BlueJ	CS1	1999	30,000,000	100	IDE	[22]
BOSS	Class Agnostic	-	-	-	Assessment	[17]
CodeHelp	Class Agnostic	2023	60	3	LLM-based	[24]
DLUnit	Computer Organization	2018	1000	4	Autograder	[23]
Dr. Scratch	Class Agnostic	2015	3,500,000	100,000	Assessment	[29]
Greenfoot	High School	2007	3,500,000	10,000	IDE	
JAAL	Data Structures and Algorithms	2020	400	1	Visualization; Autograder	[47]
Moodle Trace Generator	CS1	2020	600	1	Assessment	[41]
Mumuki	All Age	2014	100,000	250	Autograder; IDE; Assessment	[10]
OpenDSA	Data Structures and Algorithms	2013	100,000	50	Visualization; E-textbook	[9, 42]
PythonTA	CS1	2016	30,000	10	Static Analysis; Assessment	[25]
SecureCVisual	CS1	-	-	-	Visualization	[6]
SQLFE (SQL File Evaluation)	Database Systems	2019	-	-	Autograder	[51]
TigerJython	K-12	2012	500,000	1,000	IDE; Visualization	[21]

Table 1: Summary of CER-P Tools interviewed. "-" indicates that the participant did not disclosed or scaled up their tool.

4.2 Barriers

4.2.1 Lack of Incentive for Development and Deployment Beyond Publication. A barrier participants faced in the transition from a research prototype to a user-center learning tool is the magnitude of effort required to develop a system to meet academic standards while also being robust enough for widespread adoption. P7's remark captured this tension, noting:

> "You don't get to the boring bits [because] it's not the interesting research. So you're just concentrating on the interesting bit. And if you want to release it publicly and actually say in good conscience to someone at a university, 'You should use this with your course,' then it has to all work, and that's a lot of effort to get from the point where you have a proof of concept prototype that you can use yourself, that shows the interesting bit that you actually want to do your contribution, to actually then do all the boring bit to make it a properly working tool."

P5 corroborated this point and suggested that there is a *"bigger gap"* between a functioning prototype and a reliable system for wide adoption than people expected. This barrier is not merely technical but is also compounded by the lack of incentive structure in the academic system to justify the *"extra effort"* for the sake of publishing a paper. P10 attempted to estimate the magnitude of effort, stating:

> "For a paper, you need to spend X amount of effort on a prototype. And if you want to have a tool that is useful for other people, you have to spend 10X that effort."

4.2.2 Handling Educational Tool Maintenance Challenges. Participants identified maintenance, including managing user inquiries, updating documentation, and ensuring software compatibility, as a barrier to wider adoption of their tool. For example, P15 highlighted the difficulty in handling user questions or bug reports as a secondary responsibility to teaching and other duties, stating:

> "The other big issue is just around fielding user questions or bug reports. [...] This is very much a side part of the job for me, in addition to my teaching and other responsibilities. And so, definitely, that part of the maintenance, not of the software itself, but of the customer relations or the client relations, is challenging."

P7 shared their experience with the volume of feedback received following wider adoption, which included not just bug reports but also numerous feature requests:

> "Once [our tool] took off and was adopted, we got a lot of bug reports and feature requests, quite extensively."

Another aspect to maintenance developers have to consider is the need for constant updates to documentation, as described by P14 who stresses the difficulty in creating and maintaining clear, accessible instructions for both instructors and students:

> "The main challenges were around creating and then maintaining documentation for the tool, both for instructors who were considering its use and had questions, and for students who would be using it and would definitely have questions. Scalability is an issue on the user support front because while there is a documentation website, it's not amazing and needs work."

Furthermore, P2 addressed the need for regular software updates to maintain tool functionality within ever-evolving software ecosystems:

> "Any software system is never operating in a vacuum. It's in an operating system that's constantly evolving, and the libraries that it relies on are constantly evolving. [...] So, someone's got to keep updating this stuff, has to update the operating system, update the security certificates every year, and on and on."

4.2.3 Securing Sustainable Funding for Ongoing Maintenance. Participants mentioned the difficulty in securing sustainable funding for the ongoing maintenance of their tools. While there exist mechanisms to obtain funding for the initial development of tools, maintaining the tool after its development and initial funding have completed poses challenges. P16 expressed this concern, stating:

> "The problem often arises when the grant that we have for developing the software runs out. We no longer have funding to maintain the code, to improve the code, [or] to fix bugs, and then [we're] stuck with me doing that and being department chair, I don't have time to do that."

P2 echoed this sentiment, highlighting the challenge of funding for maintenance when research funding tends to prioritize novelty:

> "It's a struggle to keep it maintained. To the extent that we get new, innovative stuff and researchy ideas, we can get research funding. But it's harder to get funding for maintenance."

Due to the challenges of securing funding, P10 highlighted the personal financial commitment required to keep their tools operational:

> "At this time, I pay for [the hosting] on Google Cloud [...] €70 every month out of my pocket because it's difficult to find funding."

Acknowledging the challenges posed by limited funding, many participants acknowledged the imperfections in their CER-P tools. P9 reflected:

> "I think the major problem is that we've literally had no funding, except for like $2,000 to $3,000 that we got from donations. [...] We know there are a couple of bugs in it, but we just do not have the resources to address them."

The financial challenges associated with maintaining tools have made several developers cautious about scaling up. For example, P11 noted:

> "If I signed up ten more teachers with a thousand more students, I'm like, 'Okay, well, now I have to pay more. So how am I going to handle this?' [...] It's not a big deal, but like, somebody has to pay for it. And so, when that starts happening, then it just gets weird."

4.2.4 Lack of Developer Continuity in an Academic Setting. The challenge of managing developer churn is two-fold: maintaining consistent development practices amidst a revolving door of contributors and navigating the limited time developers can dedicate to the project. P15 captures the essence of the first challenge:

> "The number of people involved in coding, updating, fixing bugs, and various other tasks had grown quite a lot. [...] As a result, we ended up with code that had a massive variety of styles and approaches, clarity of documentation varied greatly, and so, it was a mess. We weren't in a position to treat this as a commercial project, with a hands-on manager keeping an eye on it over many years."

This issue is compounded by the time constraints that developers, often academic professionals with their own teaching and research commitments, face. P3 highlighted the realistic limits of dedicating effort to development work within the academic setting:

> "For me, it was just purely finding the time to do it myself. I mean, when you've got teaching and service requirements, finding hours to actually program is pretty rare these days."

4.2.5 Navigating the Complexity of Integration into a Modern Learning Software Ecosystem. Participants emphasized the importance of using existing protocols, such as Learning Tools Interoperability (LTI), to integrate with Learning Management Systems (LMSs) to avoid reinventing functionality. However, some participants faced technical barriers in adopting their tool with LMSs. For example, P12 explained:

> "It's required quite a bit of custom software development to integrate [my tool] into our learning management system. So, it requires some software development. It is not available as a software package that could be just integrated into a learning management system."

The goal of making tools compatible across various LMS platforms, while increasing their utility, significantly complicates the development process. P4 expressed a limited understanding of the technical requirements for such integration:

> "If we had the time or the manpower, [we] would integrate them into [LMSs] like Moodle. I don't have any idea how this would be, maybe because I don't really understand how these tools work."

Similarily, P3 suggested:

> "What I need is a step-by-step guide on how to create an LTI 1.3 app. [...] After dealing with this tool, everything I create from now on is going to be LTI compliant and just work with everything."

These barriers emphasize the value of resources like SPLICE's LTI tutorial and indicate a greater need for effectively sharing existing resources to enhance the design of educational tools.

4.2.6 ***Discussion of Barriers (RQ1).*** CER-P tools have an important place in computing education and a measurable impact on its quality. According to the developers' own estimate, their tool has had an impact on more than 37 million students. Studies on these and other CER-P tools not only suggest that they can have a positive impact on student outcomes [8, 11, 15, 16, 18, 32, 34–37, 44, 49, 50],

but they also enabled important research analysis through large-scale data collection (e.g. Blackbox [4]) and secondary research. Given these tools importance, the systemic barriers to wider adoption we present should therefore be concerning to the CER community. Participants we interviewed emphasized that the problem lies not in creating *new tools* - of which there are arguably too many - but the barriers to securing funding, developing/maintaining and promoting existing tools that have proven valuable to educators.

Our results suggest that many tool developers struggle to meet educators' needs with limited resources, and are sometimes discouraged from attracting wider audiences due to additional required resources, often in misalignment with the funding and incentive structures of academia. This aligns with findings from Hovey et al (2023) that the current structure are heavily skewed towards innovation and neglects maintenance and sustainability. This is not inherently a flaw in academic/grant funding - if its goal is to advance research, then it makes sense that it would fund research rather than the maintenance of tools - but this nonetheless presents a challenge for academically-developed tools.

To the extent that developers are able to scale up despite this barrier, this may be due to dedication, characterized by voluntary overtime and a passion of their work, or to rare fortune such as industry funding (discussed below) - a scenario not easily replicable for most. This suggests the need for more systematic support for the CER-P tool ecosystem, such that the ability for an academically-developed tool to reach wide adoption is a function of its impact and use to educators, rather than the willingness of developers to go above and beyond. In the next section, we discuss strategies that CER-P tool developers use to address many of these barriers. However, these strategies do not always work, nor do they address every challenges. Therefore, in the final section of our results, we discuss CER-P developers' views on various community actions that may address these barriers.

4.3 Practices

We found through our thematic analysis that the practices tool developers discussed fall in three categories: funding, development, and promotion/adoption. Practices are individually highlighted in each section and discussed in further detail.

4.3.1 Funding. Several participants highlight the importance of continuous support and **Leveraging industry funding** in transitioning projects into sustainable tools. For both P5 and P7, they both discussed being *"lucky to get industry funding"*. P5 reflects on how this is rare and often the compensation from tool developers is love for labor:

> "The problem for a lot of research is that they'd quite like to do what we do—turn ideas into tools and continue to maintain them. But it requires continuous funding, and a lot of tools grind to a halt because it's often just one very enthusiastic professor doing a superhuman amount of work to get the tool out there. But then, they can't sustain it long term."

P7 further highlights the impact of being able to secure continuous funding and how that can alleviate many barriers:

> "That was crucial. Without that, it would have been difficult, because we used that money to then hire a couple of part-time programmers to help with the development and support. [...] And so, that funding was necessary. I think without the funding, we could not have maintained [our tools]."

P13 shared that developing tools in a country from the Global South presented unique challenges. Initially driven by academic goals, they faced the harsh reality of obtaining funding in their context. Consequently, in order to *"try to make it sustainable"*, they adopted a *"business-level infrastructure"* and **Pursue a self-sustaining business model** by *"selling their product to private schools"*, despite not having *"intention to create a company"*. P13 further reflects:

> "And that's why we need to also scale, not only because we wanted to, not only because it was aligned with our conceptual goals, but also because it was a way of trying to make it sustainable."

4.3.2 Development. **Leverage LTI to offload LMS-style features.**

Participants emphasized the importance of using existing protocols to integrate with learning management systems in order to avoid reinventing functionality. P2 emphasized:

> "Every tool should not be an LMS. [T]ools should not have accounts, tools should not keep scores or grading data. That stuff should belong to an LMS."

Given the considerable effort required to maintain features commonly found in educational software—such as user accounts, grade books, and content management systems—many educators turn to interoperability standards like LTI, which enables educational tools to work in tangent with LMSs. This approach not only makes it easier for educational institutions to adopt these tools but also improves the user experience by centralizing educational resources and data and preventing the redundancy of LMS features in individual tools. Participants yet to adopt LTI expressed a desire to do so. P3 remarked:

> "The next version that I make won't be for [one LMS] only. I'm likely to build an LTI-compliant version that can just plug in and be used within any LMS."

Managing student developers effectively.

Due to the barrier of development continuity and time constraints, developers often have to resort to a common practice among academics: engaging students in development work as part of their learning experience.

This method, while beneficial for student education, often contributes to the inconsistency in development practices. For example, P15 highlighted:

> "I did a little bit of the initial coding, mainly with some students in my past year. I mean, that's one of the things that full-time academics find, which is once you've got beyond your PhD, then the time available for that type of activity becomes quite restricted. Unfortunate, because I like coding."

This practice of involving students and sometimes retired professors not only allows for continued development of projects but

also serves as a means to manage the workload within the limited time available to full-time academics. P2 highlighted:

> "We always had a continuous stream of good graduate students to do the next generation of that as a research project, more or less and to do maintenance on the side as as they're going along. And you know I've been fortunate that that we've been always been able to find the next person when someone graduates."

4.3.3 Promotion & Adoption. **Bundle teaching materials.**

Participants have identified that beyond the functionality of the tools themselves, it is the availability of teaching materials that help facilitate their integration in the classroom. Developing textbooks and structured curriculums that complement these CER-P tools emerged as a key strategy for encouraging their use.

P2 highlighted the importance of instructional units over isolated algorithm visualizations:

> "the big lesson we learned is that people don't want isolated algorithm visualizations; they need units of instruction. They don't want just a visualization of a quicksort; they want a quicksort unit, or maybe a sorting unit, or maybe a data structures course."

Furthermore, some participants initial believed that the innovative nature of their tools alone would drive adoption.. However, they later recognized that the presence of well-crafted teaching materials was actually key to increasing tool adoption. P7 shared:

> "At first, I thought, I built this tool, and was fully convinced that this environment is better than other environments that were out there. And I thought, you just put it out there, and people will see that it's better, and they will start adopting it, right? And that happened, but only to a limited degree. So, some people adopted it, but it wasn't really very big. What I understood later is that what drives adoption actually is teaching material."

P7 further explained that potential users are often *"lecturers, professors, and teachers"* who are pressed for time and resources and just prefer to be provided with guidance on how to incorporate these tools into their teaching. P7 reflected:

> "in retrospect, that I hadn't realized it at the beginning that the tool itself achieves very little. It's actually the teaching. If you show a teacher that they can teach differently, that's what gets interest, and the tool is just a means to make that happen, to make it possible."

Interestingly, P11 mentioned that, although their initial goal was to help educators develop customized content for their classes using their tool, they found that providing a complete sample curriculum actually drove adoption. P11 shared his experience:

> "I built the curriculum for this class, and now it's your turn to teach it. And hey, look! All the assignments are done if you just use this tool. And so they did, and many of them didn't really [create their own content]."

These insights suggest that the availability of ready-to-use teaching materials can significantly reduce the barrier to adoption since users of CER-P tools are often educators looking for effective and efficient resources to integrate into their course.

Active promotion through networks. This practice highlights the importance of *actively engaging* with the computing education community through various channels like conferences, workshops, and existing network. One participant shared their experience with utilizing SIGCSE, a popular conference in computing education, as a platform for promoting their tool. P7 mentioned:

> "SIGCSE was probably the most important conference for us to advertise at. We conducted a lot of workshops at SIGCSE, just because it's very large. There are a lot of people there, and many of them are actually teachers who are looking for software they can use."

P7 further highlighted that it was difficult "to pinpoint which activities had the main impact" when it came to promoting, but suggested it was rather a combination of providing teaching material and presentation that drove adoption. P7 reflected:

> "I think it wasn't just the book that drove adoption. The book certainly helped because it was there, and people could pick it up. But the initial interest came from our presentations."

In addition to these strategies, some participants leverage their existing networks, often their institutional colleagues, to spread the word about their tools. P6 described a more localized approach:

> "Those who have directly used it with students so far are two other faculty members in my department here. [...] I just ask them, "Hey, do you want to use this?" We're all like friends; we're in the same little office suite. So it was relatively easy."

However, not all participants actively pursued promotion as some raised concerns of scalability of tool adoption. For example, P11 mentioned they *"haven't really advertised"*, due to uncertainty over managing broader use without charging for the tool. P11 elaborated:

> "It's been people that were in my department, a friend of a friend, that kind of thing, and it just hasn't grown. Also, I don't charge for it. So if I get 100 people involved, I don't know how I'd pay for it."

Ease of Use: Being in the browser.

Developers have identified that designing their tools to run directly in the browser enhances their ease of use and adoption. For P10, they attributed it to their popularity of their tool, stateing:

> "P10: One of the decisions that made [my tool] quite popular is that it's easy to use. You only need a web browser, no installation required at all."

This approach reduces barrier to entry and by simplifying it for the user experience, P10 further highlighted that it *"makes it as easy as possible for the end users and in the end, it's why it had so much impact"*.

However, one participant noted that transitioning an application to the web, especially for a tool originally designed as offline desktop applications, is challenging. P5, acknowledging the benefits of being in browser, explained:

> "there's not an easy route to just take [my tool] and pop it in a web browser, [...] we're just in this situation where, basically, we started as an offline Java application. And realistically, we can't change unless we just do a completely new thing from scratch."

One advantage of browser-based tools is their potential to be utilized in educational environments with limited access to traditional computers but access to tablets. P9 noted:

> "It works particularly well because, at least in [my country], many schools are using iPads, so we can cater this to students in these schools."

Acknowledging the transition challenges, one participant discuss that over time, efforts were made to simplify the process for user installation:

> "P5: So when we started out, it was "go here and install Java, and from here install our thing," and everything else is kind of included. And actually, over time, we've even removed that step. So now, we include Java in all our installers for BlueJ and Greenfoot. So it's completely self-contained. [...] you just download one file, install it, and that's it. Everything is included."

Ease of use directly impacts a tool's scalability, as highlighted by P14, who discussed their tool's minimal infrastructure:

> "there wasn't a lot of infrastructure that was required for running it. Like you can download it just like any other Python package. Therefore, infrastructure for scalability was not so much an issue."

Streamlining User Support and Simplifying Tool Infrastructure. Efforts to streamline user support and simplify tool maintenance have emerged as key practices. P7 highlighted a proactive approach to user support and emphasizes the importance of responsiveness:

> "We have from the beginning and still going today, done user support. So, we have a user support email. You know, every email we get? We answer within 24 hours. And we get quite a few, you know. So, we have about 2.5 million users per year using the system. So, we get, you know, 7 emails a week. Especially when the term starts in the Western Hemisphere, we get in the order of one or two emails per day, and someone has to just sit there and answer them, you know. So, that is, it's actually work that gives you no real credit. But we have put a lot of work into supporting the users and making it have a sort of professional level of quality and support."

Meanwhile, P6 discussed strategies for keeping tool maintenance manageable, particularly for solo developers with limited funding, by leveraging external services for common functionalities such as authentication:

> "the main thing is trying to keep it simple for a couple of reasons. One, so that it's manageable for me as a solo developer, and two, so that it stays lightweight, and I don't have needs for crazy hardware, databases, or whatever. Part of that strategy has been, for example, avoiding the management of user logins and authentication on my own"

P14 emphasized the importance of clarity in documentation as a practice, especially when delegating updates to undergraduates in writing user-friendly guides, stating:

> "So, trying to make the documentation website as easy to understand as possible. What I find is that because I work with a lot of students on [my tool], and sometimes they fix bugs, and sometimes they add new features, when they add new features, I also have them update the documentation page. But what I find is that the students I work with, despite being mostly excellent programmers and doing everything on the technical front really well, they aren't as strong in the documentation side, in writing English prose that explains how to use whatever feature they're adding in a way that's appropriate for instructors and students. I don't blame them as much, it's a different skill set."

4.3.4 ***Discussion of Practices (RQ2).*** The results indicate that external funding, such as industry support, allows certain tools to overcome the barriers mentioned above. However, by developers' own admission, industry funding is often "lucky" and difficult strategy to replicate. Other developers end up charging educators for their tools, even if this was not the developer's original intention. As discussed in the prior section, the challenge of funding for maintenance, improvement, and scaling of CER-P tools may require more systematic action by the CER community, rather than innovation by individual developers. For example, it may be helpful to normalize as a community that high-quality CER-P tools developed by academics without a profit motive may still require educational institutions to pay for them, as was the case for P13. Since many CER-P developers may have no wish to run a side business, there may be value in services that facilitate the financial management of tool dissemination (e.g., analogous to how Teachers Pay Teachers[46] facilitates teacher-to-teacher payments for instructional resources). Alternatively, there may be value in additional community grants, such as the SIGCSE special projects grant [43] to fund outstanding CER-P tools.

Some developers highlighted the advantages of utilizing interoperability standards, like LTI, to delegate specific Learning Management System (LMS)-style functionalities. This approach not only streamlines the integration process but also opens up further opportunities for enhancement. A common challenge identified, however, is the lack of knowledge on achieving LTI compliance for their tools. This situation underscores the importance of improving dissemination and understanding of existing educational resources and support mechanisms. For instance, resources such as[1] already provide valuable guidance on this front. This suggest the importance of sharing successful strategies among developers, as solutions applied by some could address challenges faced by others and by encouraging an environment of shared knowledge and experiences, we could mitigate common barriers to adoption and scaling

[1] https://cssplice.github.io/lti.html

of CER-P tools. Developers shared several strategies that were effective for wider adoption for their tools. These strategies included active promotion at conferences or through existing networks, and bundling tools with teaching materials to ease the adoption process for educators. In the following section, we explore potential community actions directly with CER-P tool developers to ask their thoughts and propose subsequent steps for the CER community to enhance collaboration and support among its members.

4.4 Community Actions

In this section, we explore ideas that could support broader adoption and scalability of CER-P tools. We ask participant about three community actions, as previously mentioned in Section 3: (1) CS Community of tool developers: what might CS community of tool develop look like? , (2) CS Tool Repository: what might CS repository of tool develop look like?, and (3) Software Library of educational tools: What are common tasks that you think you and others might benefit from having libraries for when developing your tool? Respondents engaged most with the first two questions, so we focus on those here.

4.4.1 CS repository. In our semi-structured interviews, we ask participants about the potential benefits a CS repository could offer to tool developers like themselves. Participants identified several key features that would enhance the value of the repository for its potential user base. To facilitate a structured discussion, we organized the findings based on use cases: (1) for instructors and (2) for researchers.

For Instructors. Participants emphasized the importance of a repository that provides a clear *"description of the functionality"*. For instance, (P7) highlighted three questions the repository should answer: *"Are these similar to other tools, what are the differences, and what's the scope?"* Such information aids instructors in making informed decisions. Furthermore, the inclusion of a live demo, as P12 articulated, enables instructors to *"try the tool themselves and decide right away whether it suits their course or their type of instruction."*

Furthermore, the capacity to evaluate a tool's effectiveness and usability was highlighted as another essential feature of a CS repository. P7 articulated the value of incorporating:

> "user comments, reviews, or some kind of commentary comparing the tools and providing insights about them, so that a potential adopter can make an informed decision about whether it meets their needs."

P7 further highlighted the value of user feedback compared to evaluating the quality of a tool from the developer's perspective:

> "typically what I would trust most [are] reviews from existing users. Just peers telling me, 'I used this, it was a pain in the neck, you know, and nothing worked, and the configuration always broke down,' or they say, 'It was fantastic, you know, it really improved my course' [...] You don't get that from the developers themselves, because every developer is convinced that their own tool is great."

However, some participants expressed skepticism about the value of the repository for educators. Reflecting on their experiences with existing repositories of educational and computer science materials, P6 mentioned:

> "I never know exactly how valuable those sorts of things will be. When I think about the existing repositories of education materials and CS materials, I see they get some traffic. But it's really difficult, because I think the quality varies so much. They're not super well curated, and so they don't end up being as valuable as you might expect. The same thing might happen with tools."

They also highlighted the challenge of discovery within these repositories and pointed out that educators may not seek specific tools without prior knowledge of their existence. P6 further articulated:

> "educators don't know what [they are] looking for until [they] even know it exists [...] just having it available to be searched for isn't going to do much."

For Researchers. Participants discussed how a repository can help prevent duplication of research and support the reproducibility of studies. P12 mentioned the benefits of accessible interactive exercises and research tools, stating:

> "Our institution, as well as our research team, would benefits from them. Yes, maybe because if there are more interactive exercises and research tools, we might also use them in a totally new research setup. And I see that if the tools are more available, that would also benefit the replicability of some studies. Because then, you can collect similar research data with the same tool in many institutions. So, this could enhance the validity of the results."

P7 suggested that, at a minimum, the repository should include *"source code of the tool"*, complete with a *"detailed README file"* that clarifies the tool's purpose, its capabilities, and how it integrates with existing learning management systems, specifying whether the tool supports standard protocols like LTI or requires custom software development.

4.4.2 CS community of tool developers. We ask participants about the potential impact of a dedicated community for tool developers within the CER community. Participants discussed numerous potential benefits and limitations.

A primary advantage identified is the opportunity it presents for tool developers to showcase their work, receive feedback, and enhance their tool's visibility. The community could serve as a important platform for developers at various stages—whether brainstorming a new idea or seeking to expand the reach of a recently completed tool.

> "for people who have an idea for a new tool, you know, and who have built a new tool, to be able to present it, to get feedback, to gain adoption, and to increase visibility"

Participants also addressed the issue of redundancy in tool development. For example, P7 observed a tendency among computer scientist is to *"write a program"* to solve a problem, often unaware of existing solutions that address similiar needs. P7 emphasized:

> "P7: "Equally important, I think, is to just try to avoid duplicating effort. A lot of people at the moment are always building tools when something very similar actually exists already. [...] that's another purpose that would be great for a community, to first of all, get people together to find out what's already there and build on it rather than reinventing the wheel.""

P5 highlighted another dimension of the community's potential: the sharing of experiential knowledge. They noted that developers often accumulate valuable insights (e.g., the practices we identified in this paper) through trial and error that, if shared, could benefit others within the community. P5 furthers noted:

> "We've probably got some knowledge built up that we don't even realize other people would benefit from, like 'We tried this in the past, and it didn't work. We tried this, and it worked very well.' And we're not really sharing it with anyone else, just because there's not really a venue to do that [...] it doesn't tend to happen at the conferences, which tend to be more about sharing educational results rather than purely design discussions."

P8 provided a concrete example of the challenges faced by developers who are not full-time professionals but still contribute to tool development:

> "So, I like to write code. But I'm not a professional developer. It's not something I do 40 hours a week. So every now and then, I'll run into a design challenge where I'm not quite sure what the best way to organize it is to make it easier to test, or easier for others to understand. It took me a couple of days to figure out how to bundle these jar files together. So, a community of CS educators who build tools—there's a ready-made community you can ask."

4.4.3 ***Discussion on Community Actions (RQ3).*** In Sections 4.2.6 and 4.3.4, we have already discussed one possible action the CER community might take to better fund the development, maintenance, and scaling of CER-P tools. Here, we discuss how CER-P tool developers viewed two other ideas: a CS tools repository and a community network.

CS Tool Repository. The effectiveness of a CS Tool Repository hinges on buy-in from the CER community. Fortunately, Blanchard et al. [3] have started the effort to develop a CS tool repository[2] that consolidates both CER tools and professional tools, marking a significant step in centralizing and organizing the wide variety of tools available for computing education. The challenge here is to demonstrate the value of the repository not just as a collection of tools but as a critical resource for educators and researchers. Our findings show that the CS Tool Repository is a promising community action solution that could streamline the approach for tool developers to promote their tool, educators to evaluate various options of tools for their specific needs, and researchers to avoid duplicating efforts. Additionally, the repository could address several barriers identified in RQ1 that the strategies discussed in RQ2 could not address. The repository could provide a dedicated platform for showcasing CER-P tools that demonstrated significant impact on student outcomes, and encourage developers to adopt existing tools rather than develop from scratch. Our findings also highlight key attributes that would benefit potential users, such as user reviews and live demo, that would assist educators in navigate an over saturated market of tools.

However, convincing potential users to contribute to and use the repository regularly would require clear communication of its benefits and ongoing engagement strategies. For the repository to be effective, it must be more than a passive archive; it needs active engagement from its users. This includes regular updates, reviews of tools, and contributions from the community. A lack of engagement could result in outdated information and diminish the relevance and usefulness of the repository, as highlighted by P6. Despite these efforts, the repository's adoption could still encounter challenges, such as resistance to change, lack of awareness, or uncertainty about integration into existing practices. Overcoming these obstacles might involve active promotion at academic conferences, hosting integration support workshops, and highlighting success stories that illustrate the repository's positive impact on education and research.

Community Network. Establishing a community network for CER-P tool developers involves creating an environment that supports active participation, collaboration, and the exchange of knowledge. Various platforms, such as workshops, "birds of a feather" gatherings, and asynchronous communication tools such as Slack or email, can cater to these needs, offering different levels of engagement. A primary benefit identified by participants is the platform's capacity for tool developers to showcase their work, receive feedback, and enhance their tool's visibility. This suggests the need for a platform that supports developers at various stages, from brainstorming new ideas to expanding the reach of recently completed tools. Similar to the CS Tool Repository, this community network requires buy-in from diverse stakeholders, including educators, developers, and researchers. Clearly defining the network's goals and benefits is crucial; otherwise, it risks failure due to perceived irrelevance.

A potential benefit of this community network could allow the developer to exchange valuable insights and address current or potential barriers in real-time, as more experienced developers can share their design choices, development strategies, and the decision-making processes behind specific tool features. Insights gained through trial and error, which could significantly benefit many tool developers, are often not shared due to the lack of a suitable platform. This suggests the need for a venue dedicated to the exchange of design discussions and practical experiences, different from traditional conference settings that focus more on educational outcomes. The community's role could help minimize duplicated efforts by facilitating knowledge sharing about existing tools and encourage developers to build upon what is already there instead of reinventing the wheel. However, maintaining engagement over time presents a significant challenge. Interest may wane if members do not perceive immediate benefits or if the activities offered by the community become monotonous or irrelevant. Strategies to keep the community active and engaged include regular updates,

[2]https://csed-tools.github.io/

highlighted discussions, and the showcasing of successful collaborations, all of which are essential for ensuring the sustainable long-term impact of the community network.

5 LIMITATIONS

Although our study offers valuable insights, it's important to acknowledge several limitations that affect the internal and external validity of our findings. Firstly, the decision to focus on CER-P tools developed within the past five years, while intentional to assess modern tools, may have inadvertently narrowed the scope of our research. The inclusion of tools cited by [3], which extends beyond our initial timeframe, was aimed at broadening our perspective, but also introduced variability in the technological and pedagogical contexts of the tools considered. For instance, tools created several years ago may not have been designed with current technologies in mind, such as web-based accessibility or compatibility with modern LMSs. Thus, older tools may have faced barriers that are no longer relevant for the development of new CER-Programming tools today.

Secondly, our data collection methods involved choosing participants who agreed to participate in the interview section from the surveys, which resulted in limited control over participant selection. This likely introduce potential selection bias and resulted in a sample skewed towards developers who are more engaged or faced significant challenges with tool adoption. The effects of this bias on our results are uncertain, possibly highlighting successes or barriers in a way not fully reflect the wider community of CER-P tool developer.

Furthermore, we only asked participants about a limited set of community actions, such as the Community Network of Tool Developers (presented in Section 4.4), rather than inviting an open-ended discussion of useful community actions. We make no claim that these are the only, or best, actions the community could take, but our findings inform *how* we might go about building community or creating a CS tools repository. We also acknowledge that, as members of the SPLICE project (Standards, Protocols, and Learning Infrastructure for Computing Education) [3], the authors are actively working on creating such infrastructure to support CER-P tool developers. As such, our findings may reflect our bias towards these projects. Fully understanding the potential benefits and challenges of these ideas requires further investigation, and future work should explore the feasibility, structure, and impact of such an initiative on the broader adoption and scalability of CER-P tools.

Our reliance on self-reported data introduces a level of subjectivity into our findings. The focus of our study was on developers' perspectives, which, while critical, is just one angle of the whole process of tool adoption in educational settings. It's crucial to recognize that other stakeholders, including instructors, educational institutions, and students, may offer differing insights into the barriers to adoption and the impacts of these tools on learning outcomes. Future research should aim to incorporate these varied perspectives to provide a better understanding of the factors influencing the success and impact of CER-P tools in educational contexts.

[3]https://cssplice.org

6 CONCLUSION

Despite these limitations, our findings offer valuable insight into the perspective of developers on the dissemination and adoption of the CER-P tools. Firstly, our work found that despite the recognized importance of CER-P tools in computing education, systemic barriers significantly hinder their wider adoption. The key issues identified include challenges not in creating new tools, but in securing funding, development, maintenance, and promotion of existing tools that are valuable to educators. Additionally, many tool developers face difficulties in meeting educators' needs due to limited resources, further compounded by the misalignment of funding and academic incentive structures that are crucial for the maintenance of tool.

Secondly, our study details some key practices the participants employed that help enable broader adoption, such as creating units of curriculum alongside their tool, leveraging interoperability standards and actively promoting at conferences. While our work found that external funding, like industry support, enables some tools to surpass the barriers previously identified, such funding is often inconsistent and hard to replicate. Future work should look into developing sustainable financial services for software tool developed in academia. Lastly, our work found that alongside previously discussed funding strategies, CER-P tool developers see significant potential in two community actions initiative: establishing a CS Tool Repository and a Community Network. Future work should look into the strategy to enhance engagement and value perception of these initiatives from the wider community.

Our study demonstrates the need for a more systematic support for CER-P tools within the CER community. The capacity of developers to scale up, often fueled by a mix of dedication, voluntary overtime, and the rare chance of industry funding, highlights factors that are not easily replicated or sustained. This calls for a shift in the way academically developed tools are valued and supported. As a CER community that has benefited massively from CER-P tools, it is vital that we move towards a direction where the intrinsic value of educational tools is acknowledged and promoted based on impact and utility rather than the efforts of tool developer to go beyond.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant #2213792.

REFERENCES

[1] Lecia Barker, Christopher Hovey, and Jane Gruning. 2015. What Influences CS Faculty to Adopt Teaching Practices? *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015), 604–609. https://doi.org/10.1145/2676723.2677282

[2] Michael Berry and Michael Kölling. 2014. The State of Play: A Notional Machine for Learning Programming. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (Uppsala, Sweden) *(ITiCSE '14)*. Association for Computing Machinery, New York, NY, USA, 21–26. https://doi.org/10.1145/2591708.2591721

[3] Jeremiah Blanchard, John R. Hott, Vincent Berry, Rebecca Carroll, Bob Edmison, Richard Glassey, Oscar Karnalim, Brian Plancher, and Seán Russell. 2022. Stop Reinventing the Wheel! Promoting Community Software in Computing Education. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* 22 (dec 2022), 261–292. https://doi.org/10.1145/3571785.3574129

[4] Neil CC Brown, Amjad Altadmri, Sue Sentance, and Michael Kölling. 2018. Blackbox, five years on: An evaluation of a large-scale programming data collection

project. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 196–204.

[5] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, Jaime Urquiza, Arto Vihavainen, and Michael Wollowski. 2014. Increasing Adoption of Smart Learning Content for Computer Science Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (Uppsala, Sweden) *(ITiCSE-WGR '14)*. Association for Computing Machinery, New York, NY, USA, 31–57. https://doi.org/10.1145/2713609.2713611

[6] Steve Carr and Jean Mayo. 2020. SecureCvisual: Visualization and Analysis for C Code Security. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. ACM. https://doi.org/10.1145/3328778.3372540

[7] Wanda Dann, Dennis Cosgrove, Don Slater, Dave Culyba, and Steve Cooper. 2012. Mediated Transfer: Alice 3 to Java. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (Raleigh, North Carolina, USA) *(SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 141–146. https://doi.org/10.1145/2157136.2157180

[8] Matthew Heinsen Egan and Chris McDonald. 2021. An evaluation of SeeC: a tool designed to assist novice C programmers with program understanding and debugging. *Computer Science Education* 31, 3 (2021), 340–373. https://doi.org/10.1080/08993408.2020.1777034

[9] Eric Fouh, Ville Karavirta, Daniel A Breakiron, Sally Hamouda, Simin Hall, Thomas L Naps, and Clifford A Shaffer. 2014. Design and architecture of an interactive eTextbook–The OpenDSA system. *Science of computer programming* 88 (2014), 22–40.

[10] Marcos J. Gomez, Marco Moresi, and Luciana Benotti. 2019. Text-based Programming in Elementary School: A Comparative Study of Programming Abilities in Children with and without Block-based Experience. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*. ACM. https://doi.org/10.1145/3304221.3319734

[11] Philip Guo. 2021. Ten Million Users and Ten Years Later: Python Tutor's Design Guidelines for Building Scalable and Sustainable Research Software in Academia. *UIST 2021 - Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology* (oct 2021), 1235–1251. https://doi.org/10.1145/3472749.3474819

[12] Lassi Haaranen, Lukas Ahrenberg, and Arto Hellas. 2023. Decades of Striving for Pedagogical and Technological Alignment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (<conf-loc>, <city>Koli</city>, <country>Finland</country>, </conf-loc>) *(Koli Calling '23)*. Association for Computing Machinery, New York, NY, USA, Article 23, 8 pages. https://doi.org/10.1145/3631802.3631809

[13] David Hammer and Leema K Berland. 2014. Confusing claims for data: A critique of common practices for presenting qualitative research on learning. *Journal of the Learning Sciences* 23, 1 (2014), 37–46.

[14] Christopher Lynnly Hovey, David P. Bunde, Zack Butler, and Cynthia Taylor. 2023. How Do i Get People to Use My Ideas? Lessons from Successful Innovators in CS Education. *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education* 1 (mar 2023), 841–847. https://doi.org/10.1145/3545945.3569779

[15] Ryosuke Ishizue, Kazunori Sakamoto, Hironori Washizaki, and Yoshiaki Fukazawa. 2018. PVC: Visualizing C Programs on Web Browsers for Novices. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 245–250. https://doi.org/10.1145/3159450.3159566

[16] Masita Jalil, Shahrul Azman Noah, and Sufian Idris. 2010. Evaluating the Effectiveness of a Pattern Application Support Tool for Novices. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education* (Bilkent, Ankara, Turkey) *(ITiCSE '10)*. Association for Computing Machinery, New York, NY, USA, 239–243. https://doi.org/10.1145/1822090.1822158

[17] Mike Joy, Nathan Griffiths, and Russell Boyatt. 2005. The boss online submission and assessment system. *Journal on Educational Resources in Computing (JERIC)* 5, 3 (2005), 2–es.

[18] Erkki Kaila, Teemu Rajala, Mikko-Jussi Laakso, and Tapio Salakoski. 2010. Effects of Course-Long Use of a Program Visualization Tool. In *Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103* (Brisbane, Australia) *(ACE '10)*. Australian Computer Society, Inc., AUS, 97–106.

[19] Oscar Karnalim and Simon. 2021. Explanation in Code Similarity Investigation. *IEEE Access* 9 (2021), 59935–59948. https://doi.org/10.1109/ACCESS.2021.3073703

[20] Tobias Kohn and Bill Manaris. 2020. Tell Me What's Wrong: A Python IDE with Error Messages. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1054–1060. https://doi.org/10.1145/3328778.3366920

[21] Tobias Kohn and Bill Manaris. 2020. Tell Me What's Wrong: A Python IDE with Error Messages. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1054–1060.

[22] Michael Kölling, Bruce Quig, Andrew Patterson, and John Rosenberg. 2003. The BlueJ system and its pedagogy. *Computer Science Education* 13, 4 (2003), 249–268.

[23] Zachary Kurmas. 2023. DLUnit: A Unit Testing Framework for Simulated Digital Logic Circuits. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–6.

[24] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2023. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (Koli Calling '23)*. ACM. https://doi.org/10.1145/3631802.3631830

[25] David Liu, Jonathan Calver, and Michelle Craig. 2024. A Static Analysis Tool in CS1: Student Usage and Perceptions of PythonTA. In *Proceedings of the 26th Australasian Computing Education Conference*. 172–181.

[26] Nuno Macedo, Alcino Cunha, José Pereira, Renato Carvalho, Ricardo Silva, Ana C R Paiva, Miguel Sozinho Ramalho, and Daniel Silva. 2021. Experiences on teaching alloy with an automated assessment platform. *Science of Computer Programming* 211 (2021), 102690. https://doi.org/10.1016/j.scico.2021.102690

[27] Nuno Macedo, Alcino Cunha, José Pereira, Renato Carvalho, Ricardo Silva, Ana C R Paiva, Miguel Sozinho Ramalho, and Daniel Silva. 2021. Experiences on teaching alloy with an automated assessment platform. *Science of Computer Programming* 211 (2021), 102690. https://doi.org/10.1016/j.scico.2021.102690

[28] Jane Mills, Ann Bonner, and Karen Francis. 2006. The development of constructivist grounded theory. *International journal of qualitative methods* 5, 1 (2006), 25–35.

[29] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[30] Lijun Ni. 2009. What makes CS teachers change? factors influencing CS teachers' adoption of curriculum innovations. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA) *(SIGCSE '09)*. Association for Computing Machinery, New York, NY, USA, 544–548. https://doi.org/10.1145/1508865.1509051

[31] Rohan Padhye, Koushik Sen, and Paul N. Hilfinger. 2019. ChocoPy: A Programming Language for Compilers Courses. In *Proceedings of the 2019 ACM SIGPLAN Symposium on SPLASH-E* (Athens, Greece) *(SPLASH-E 2019)*. Association for Computing Machinery, New York, NY, USA, 41–45. https://doi.org/10.1145/3358711.3361627

[32] James H. Paterson, John Haddow, and Ka Fai Cheng. 2008. PatternCoder: A Programming Support Tool for Learning Binary Class Associations and Design Patterns. In *Proceedings of the 8th International Conference on Computing Education Research* (Koli, Finland) *(Koli '08)*. Association for Computing Machinery, New York, NY, USA, 96–100. https://doi.org/10.1145/1595356.1595375

[33] Lutz Prechelt and Guido Malpohl. 2003. Finding Plagiarisms among a Set of Programs with JPlag. *Journal of Universal Computer Science* 8 (2003).

[34] Thomas Price, Rui Zhi, and Tiffany Barnes. 2017. Evaluation of a Data-Driven Feedback Algorithm for Open-Ended Programming. *International Educational Data Mining Society* (2017), 192–197.

[35] Thomas W. Price, Neil C.C. Brown, Dragan Lipovac, Tiffany Barnes, and Michael Kölling. 2016. Evaluation of a Frame-based Programming Editor. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, 33–42. https://doi.org/10.1145/2960310.2960319

[36] Yizhou Qian and James D Lehman. 2019. Using Targeted Feedback to Address Common Student Misconceptions in Introductory Programming: A Data-Driven Approach. *SAGE Open* 9, 4 (2019), 2158244019885136. https://doi.org/10.1177/2158244019885136

[37] Kelly Rivers. 2017. *Automated data-driven hint generation for learning programming*. Ph. D. Dissertation. Carnegie Mellon University.

[38] Susan H. Rodger, Julian Genkins, Ian McMahon, and Peggy Li. 2013. Increasing the Experimentation of Theoretical Computer Science with New Features in JFLAP. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) *(ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 351. https://doi.org/10.1145/2462476.2466521

[39] Guido Rößling, Markus Schüler, and Bernd Freisleben. 2000. The ANIMAL Algorithm Animation Tool. *SIGCSE Bull.* 32, 3 (jul 2000), 37–40. https://doi.org/10.1145/353519.343069

[40] Pálma Rozalia Osztián, Zoltán Kátai, and Erika Osztián. 2020. Algorithm Visualization Environments: Degree of interactivity as an influence on student-learning. In *2020 IEEE Frontiers in Education Conference (FIE)*. 1–8. https://doi.org/10.1109/FIE44824.2020.9273892

[41] Seán Russell. 2022. Automated code tracing exercises for cs1. In *Proceedings of the 6th Conference on Computing Education Practice*. 13–16.

[42] Clifford A Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L Naps. 2011. OpenDSA: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on computing education research*. 112–117.

[43] SIGCSESpecialPrograms 2023. *SIGCSE Special Programs*. Retrieved August 17, 2023 from https://sigcse.org/programs/special/

[44] John Stamper, Michael Eagle, Tiffany Barnes, and Marvin Croy. 2013. Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education* 22, 1-2 (2013), 3–17.

[45] Cynthia Taylor, Jaime Spacco, David P. Bunde, Zack Butler, Heather Bort, Christopher Lynnly Hovey, Francesco Maiorana, and Thomas Zeume. 2018. Propagating the adoption of CS educational innovations. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018 Companion)*. Association for Computing Machinery, New York, NY, USA, 217–235. https://doi.org/10.1145/3293881.3295785

[46] Teacherspayteachers 2023. *Teacherspayteachers*. Retrieved August 17, 2023 from https://www.teacherspayteachers.com/

[47] Artturi Tilanterä, Giacomo Mariani, Ari Korhonen, and Otto Seppälä. 2021. Towards a json-based algorithm animation language. In *2021 working conference on software visualization (vissoft)*. IEEE, 135–139.

[48] Dwayne Towell and Brent Reeves. 2010. From Walls to Steps: Using online automatic homework checking tools to improve learning in introductory programming courses. (2010).

[49] Kelsey Van Haaster and Dianne Hagan. 2004. Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool. *Issues in Informing Science & Information Technology* 1 (2004).

[50] Paul J. Wagner. 2020. The SQL File Evaluation (SQLFE) Tool: A Flexible and Extendible System for Evaluation of SQL Queries. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1334. https://doi.org/10.1145/3328778.3372599

[51] Paul J Wagner. 2020. The sql file evaluation (sqlfe) tool: A flexible and extendible system for evaluation of sql queries. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1334–1334.