End-User Programmers Repurposing End-User Programming Tools to Foster Diversity in Adult End-User Programming Education

Sean Kross
UC San Diego
La Jolla, CA, USA
seankross@ucsd.edu

Philip J. Guo
UC San Diego
La Jolla, CA, USA
pg@ucsd.edu

***Abstract*—Efforts to improve diversity in computing have mostly focused on K-12 and university student populations, so there is a lack of research on how to provide these benefits to adults who are not in school. To address this knowledge gap, we present a case study of how a nine-member team of end-user programmers designed an educational program to bring job-relevant computing skills to adult populations that have traditionally not been reached by existing efforts. This team conceived, implemented, and delivered Cloud Based Data Science (CBDS), a data science course designed for adults in their local community in historically marginalized groups that are underrepresented in computing fields. Notably, nobody on the course development team was a full-time educator or software engineer. To reduce the amount of time and cost required to launch their program, they repurposed end-user programming skills and tools from their professions, such as data-analytic programming and reproducible scientific research workflows. This case study demonstrates how the spirit of end-user programming can be a vehicle to drive social change through grassroots efforts.**

***Index Terms*—diversity in computing, end-user programming, data science**

I. Introduction

A widely-acknowledged deficit in computing fields is the lack of historically underrepresented groups on teams that build software and make engineering decisions [1], [2]. This deficit of perspectives is especially impactful considering how algorithmic-driven decision-making has become a fundamental part of modern life. Algorithms determine who is approved for bank loans [3], which job applications are considered for job openings [4], and what plan of care certain patients end up receiving [5]. Data-driven systems have also re-enforced racial biases [2] and denied essential social services to members of historically marginalized groups [6]. The reasons for these failings are multifaceted, but they are compounded when contributions from the people who would be most affected are excluded from the system design process [2], [6].

In recent years both researchers and community organizations have addressed these issues by diversifying the population of students who choose to study computing. For instance, academic projects such as Storytelling Alice [7], Scratch [8], and App Inventor [9] have fostered more inclusive programming communities at the K-12 level, especially in middle and high schools. Nonprofits such as Code2040 [10], Girls Who Code [11], and Black Girls Code [12] strive to improve both gender and racial diversity amongst K-12 students interested in computing. At the university level, research-backed curricula such as media computation [13] and diverse paths into computing [14], [15] have made advances in the proportion of computing majors from underrepresented groups. In addition, college scholarships and mentoring programs have helped retain such students as they advance through school. However, the majority of such efforts target K-12 and university students, so there is a lack of knowledge about how to provide these benefits to adults who are *not in school*.

To address this gap, this paper presents a case study of a team of academic research scientists who partnered with a local community organization to teach data science to adults living in a high-poverty area of a large U.S. city. The typical student in this program is an adult member of an underrepresented and marginalized group who did not complete high school; they may have grown up in foster care or may have experienced extended periods of unemployment or homelessness. The program's goal is to equip these adults with basic data science skills required to get entry-level jobs doing tasks such as spreadsheet data entry, data cleaning, wrangling, and validation. These types of data-oriented jobs offer an on-ramp into computing careers while being more within their reach than full-time software developer positions, which require much more extensive training.

To implement this grassroots initiative on a short time frame with a small budget, the team had to perform end-user programming [16] to repurpose existing tools from their research workflows and to create new ad-hoc tools to support course development. Specifically, they developed text-based programmatic workflows based on R Markdown computational notebooks [17] that they use in their research lab.

Our study is the first, to our knowledge, to analyze how a team of end-user programmers (i.e., academic research scientists) applied the philosophy of end-user programming (i.e., repurposing/building software tools for personal use) to diversify end-user programming (i.e., data science) education

978-1-7281-0810-0/19/$31.00

to reach traditionally underrepresented groups. While this paper reports a single case study, we believe its findings have generalizable research value to the field of end-user programming. Specifically, it advances the idea that *end-user programming can be a vehicle for positive social change by enabling a small team of non-specialists to repurpose software tools to serve their user population quickly and at low cost*.

This paper makes the following contributions:

- Findings from a case study of a nine-member research lab on how they performed end-user programming to repurpose tools from their research to foster education.
- Implications for end-user programming research and practice, especially related to social good, diversity in computing, and broadening educational opportunities.

II. Related Work

Our study extends and brings together prior work in two main areas: end-user programming and diversity in computing.

A. End-User Programming

End-user programming is commonly defined as the act of programming as a means to personal ends rather than for producing software artifacts for widespread public use [16], [18], [19]. This definition encompasses a wide variety of personas, ranging from professional software engineers writing throwaway prototype code to teachers writing spreadsheet macros to track grades. (Many prefer to use the term "end-user programm*ing*" to focus on the activity [16], but for notational simplicity we will use "end-user programm***er***" to refer to people who frequently perform end-user programming.)

In this paper we study academic research scientists who perform end-user programming by writing code in the R language. Prior work has studied how scientists code for their research in a range of settings, including high-performance computing [20] and across the physical and social sciences [21]–[23]. In recent years, groups have studied how researchers use Jupyter notebooks [24]–[26] in their end-user programming workflows. However, whereas prior work has focused on scientists writing code to support their own research, our study is unique in showing how they *repurpose those skills* to create an ad-hoc educational platform outside of their core research.

Besides being end-user programmers, our study participants are also creating an educational initiative to train future end-user programmers: data scientists who use code as a means to produce data-driven insights. Longstanding efforts such as Software Carpentry [27] and Data Carpentry [28] have trained data scientists in academia through volunteer-run university workshops. We have previously surveyed a broad range of data science teaching programs across universities, coding bootcamps, and industry settings [29]. While many of these efforts strive to provide an inclusive and welcoming environment, they still mainly target graduate students and working professionals who often already have advanced degrees. In contrast, the team that we study is working with a local community organization to provide free data science training to adults in underrepresented groups who often did not even finish high school. We know of no prior academic research on end-user programming education being extended to underserved populations like the one that we study.

B. Diversity in Computing

Diversity in computing has been a longstanding interest in computing education research, and it is also this year's VL/HCC special emphasis topic [30]. To our knowledge, the majority of efforts around this topic have been for students in K-12 and university settings. In contrast, we study a program to teach computing to adults who are not in school settings.

At the K-12 level (elementary, middle, and high schools), researchers have developed domain-specific programming environments to broaden interest in computing amongst traditionally underrepresented groups. For instance, Storytelling Alice [7] focused on engaging female middle school students, and Scratch [8] was deployed to after-school programs to foster computing interest amongst low-income African American and Latinx youths from 8 to 18 years old [31]. Beyond programming, the Glitch Game Tester project [32]–[34] hired African American high school students as game testers, which sparked their interest in computing careers. Project Rise Up 4 CS [35] used in-person mentorship and financial incentives to encourage African American high school students to succeed on the annual AP Computer Science exam.

At the university level, research-based diversity initiatives have focused on two fronts: curriculum design and activities inside the classroom. On the curricular front, alternative pathways into computer science [14], more flexible threads of courses for different interests [15], and service learning opportunities [36] have improved diversity in computing majors. Within the classroom, pair programming, peer instruction [37], [38], and media computation activities [13], [39] have improved retention for students from underrepresented groups.

In contrast to K-12 and university initiatives, we study the development of a free computing education program targeted at adults who do not have access to formal schooling. Prior research on adult learning of computing has studied how working adults take online courses [40], how older adults over 60 years old [41] learn to code on their own, and how end-user programmers learn to code on the job [42]–[44]. However, these adults often already have higher education and plentiful access to technology. Despite these differences in learner demographics, the educational program that we study addresses some of the same challenges of adult education that prior work found, most notably lack of time given other life responsibilities and feelings of isolation due to lack of in-person support. Finally, our study is unique in showing how a team attempted to address diversity in computing by *using the tools of end-user programming at their disposal* to develop an adult data science education program.

III. Methods

We performed a case study of the development process of *Cloud Based Data Science* (CBDS), a free online course described in Table I. The goal of CBDS is to teach basic

data science skills using spreadsheets and the R language in order to prepare students to obtain jobs as entry-level data scientists. In essence, it is training students to become end-user programmers who write code as a means to an end to clean data and produce analysis outputs.

For this case study we interviewed everyone involved in creating CBDS: eight research scientists at a large U.S. university and the project's program administrator, who was a research administrator in their lab. None of the nine team members' full-time jobs were to create educational programs or to write software; CBDS was a voluntary effort. The first author conducted all interviews (each lasting 45 to 60 minutes) using video conferencing software.

The interviews were semi-structured with questions focusing on the motivations each team member had for working on this project, their use and development of software tools during the project, and how these tools affected their interactions with other team members. Interview questions included:

- How did you first get involved in CBDS?
- What was your role in developing CBDS?
- What existing tools have you used for educational content development?
- What was your level of expertise with these particular tools before CBDS? (if they mentioned specific tools)
- Did you have to build any of your own software tools to develop CBDS? If so, which ones?
- How were development tasks distributed throughout the team?
- How did you coordinate work between team members?

The first author took notes and recorded verbatim quotations during every interview. After all interviews completed, the research team (two members) iteratively categorized them into major themes using an inductive analysis approach [45].

A. Study Design Limitations

This project was a case study of a specific team of academics at a U.S. university who attempted to develop a nontraditional educational program. Thus, we do not have large-scale replicable data and cannot claim that the CBDS team's experiences generalize to other related efforts. Also, we are relying solely on interviews and did not perform an ethnography to observe the team when CBDS was first being developed. Note that CBDS is still under active development, so many of the details are fresh on participants' minds.

Since CBDS is still in its early stages, having enrolled only around a dozen students so far, it is too early to tell the long-term outcomes of this program in terms of sustainability and impacts on its alumni. We also did not have direct access to the students and thus cannot report on their experiences. This study focuses solely on the CBDS development team.

IV. CBDS Goals: Diversify End-User Programming

We report our case study's findings by first detailing the goals of CBDS and the motivations of its volunteer development team. Then we describe their end-user programming activities throughout project development.

Module	Subject
1	Introduction to the CBDS Program
2	How to use Your Chromebook Laptop
3	How to use Web Applications
4	Organizing a Data Science Project
5	The Command-Line and Version Control
6	R Programming
7	Data Wrangling
8	Data Visualization
9	Connecting to Data Sources
10	Data Analysis
11	Communicating Analysis Results
12	Getting a Data Science Job

TABLE I
CURRICULUM OF THE CBDS (CLOUD BASED DATA SCIENCE) COURSE.

Table I shows the curriculum of CBDS, a self-paced online course that anyone can take for free. However, being free and online is nowhere near sufficient for ensuring that it is accessible to many members of underrepresented groups. Over the past decade of research into MOOCs (Massive Open Online Courses), a widely-acknowledged finding is the notable lack of diversity in who takes and benefits from them: MOOC students are mostly white or Asian males with at least college- or graduate-level degrees [46]–[48].

Interview participants P1 and P3 saw this lack of diversity firsthand since they had prior experience creating data science MOOCs. P1 explained his motivation for starting CBDS: *"Why aren't certain groups of people using our existing MOOCs? Maybe they didn't have access to hardware, they lacked prerequisite knowledge, or they were just unaware that data science was a thing."* P4 mentioned that existing courses assume prior educational experiences that exclude people without access to such opportunities: *"The problem with data science programs is that the material is pretty advanced. They're geared towards master's students."* The CBDS team believed that with a more accessible curriculum and personalized teaching approach, they could bring data science to a group that has not traditionally been reached by MOOCs. Specifically, P1's goal was to target students with a 10th-grade level of math literacy. The team also augmented CBDS with in-person support to help members of underrepresented groups enroll, remain in, and successfully complete the course.

First they worked with a local community organization to recruit potential students. To reach its target audience, the CBDS team partnered with the Historic East Baltimore Community Action Coalition (HEBCAC) [49], a nonprofit that specifically serves the historically disenfranchised low-income neighborhoods surrounding the university where the team works. HEBCAC serves a community where many residents did not complete high school, grew up in foster care, or experienced extended periods of joblessness or homelessness. HEBCAC steps in to help them complete their GED diploma (the equivalent of a U.S. high school degree), place them in jobs, or help them arrange further educational opportunities such as community college. The majority of people served by HEBCAC are African American, Hispanic, and Latinx adults. HEBCAC was a critical bridge between potential students and the CBDS team. Otherwise these students would not likely

ID	Gender	Field	Job Title	End-User Programming Experience	Created Course Content?	In-Person Tutor?
P1	M	Biostatistics	Research Lab P.I.	> 5 years	Yes	No
P2	F	Genetics	Postdoc	$1 - 5$ years	Yes	Yes
P3	M	Biostatistics	Research Scientist	> 5 years	No	No
P4	M	Biostatistics	Research Scientist	> 5 years	Yes	No
P5	F	Biostatistics	Research Scientist	> 5 years	Yes	No
P6	F	Biostatistics	Ph.D. Student	$1 - 5$ years	Yes	No
P7	F	Liberal Arts	Administrative Staff	none	No	Yes
P8	M	Economics	Postdoc	< 1 year	Yes	Yes
P9	F	Genetics	Ph.D. Student	< 1 year	Yes	No

TABLE II
BACKGROUNDS OF THE NINE MEMBERS OF THE CBDS TEAM THAT WE INTERVIEWED, ALONG WITH THEIR PRIOR END-USER PROGRAMMING EXPERIENCE AND WHETHER THEY CREATED COURSE CONTENT OR SERVED AS IN-PERSON TUTORS.

know about the existence of data science as a career path that could be within their reach.

Once students enroll, they are given a free Chromebook laptop (detailed in Section VI-A) and the opportunity to meet in-person with volunteer tutors twice per week during 90-minute office hours; P2, P7, and P8 served as tutors. Students can also ask online questions to course staff in a private Slack chat channel. Finally, to encourage retention in the program, students are paid a modest stipend for successfully completing each module in Table I; this stipend is designed to be comparable to the wage they would earn from working in the kinds of jobs that HEBCAC normally helps them obtain.

Once students finish the course, the CBDS team and HEBCAC work with them to do resume and interview preparation and to refer them to entry-level data science jobs in the area.

V. MOTIVATIONS OF CBDS DEVELOPMENT TEAM

Not only was CBDS's goal to train new end-user programmers (i.e., data scientists), its course development team also consisted of end-user programmers. Table II shows team members' backgrounds. Everyone on the team works in the same life science research group at a large U.S. research university. P1–P6 all had several years of end-user programming experience, in the form of using bioinformatics pipelines and programming as part of statistical data analysis for their research. P8 and P9 had limited programming experience before working on CBDS, while P7 had no programming experience before joining. None of the team members have a degree in computer science or experience doing professional software development. All are cisgendered (5 female, 4 male).

Why was this team motivated to create CBDS when their primary job was as research scientists? Their workplace is located in the same neighborhood served by HEBCAC, an area that has been historically disenfranchised. Decades of societal inequity has led to increased rates of poverty, which everyone on the team sees around them. Thus, all team members reported their primary motivation as wanting to create opportunities for adults in the surrounding neighborhood who could not normally afford to pay the tuition for a traditional education like that offered at their university.

P7 was closest to the target student community. She does most of the administrative work for CBDS and serves as a volunteer tutor for it. She grew up near the area served by HEBCAC, so she was very motivated to see people from her community succeed in this program: *"My personal excitement about joining in the first place was to help my people."* Besides growing up in the area, P7 felt that she was also able to relate to the students because she had only recently started learning how to code: *"I really appreciate my position in the program because I believe I am the least experienced staff member in terms of programming. So I experience the same frustrations and joys when a program crashes or when my graph turns out how I thought it would."* Also, as the only member of the team who was not on a Ph.D.-oriented research career path, she felt that students could be more honest and open with her: *"I definitely think it was a good idea to have somebody on the team who they weren't intellectually intimidated by."*

VI. END-USER REPURPOSING OF EXISTING TOOLS

Because CBDS was developed by a team of volunteer non-specialists, they needed to engage in a variety of end-user programming activities to make this program work with relatively little time and money. The first set of activities we describe here, while not "programming" per se, invoked the spirit of end-user programming by repurposing existing hardware and software to develop a data science curriculum.

A. Repurposing Low-Cost Chromebook Hardware

Keeping costs as low as possible was a major concern for the team. P1 described how prohibitively expensive it would be to build CBDS as an official university course or MOOC: *"In a traditional college setting if you had assembled several faculty to build this program it would have cost millions of dollars. We did not have that!"* Cost minimization was not just a concern in terms of development, but it was critical to the team's mission to make data science education available to underserved members of their local community. One initial obstacle they faced was simply making the technology required for doing data analytic work available to students. The students that they wanted to reach typically did not even own personal computers, or their computers were too old to install modern data science tools on: *"We wanted to reduce the cost of the hardware you need to get started. For low income folks these small costs are insurmountable."* (P3)

The team's solution was to provide a Chromebook laptop for free to every student in the program. Many Chromebooks

now cost less than $300, which is affordable compared to the typical hardware that data scientists use and well within the constraints of the seed funding provided to launch the program with the first dozen students. In addition, Chromebooks are sometimes available to check out for free from public libraries.

Each Chromebook runs Chrome OS, an operating system geared for web applications. Instead of relying on a computer with powerful hardware, students used RStudio Cloud [50], a free data science run-time environment for the R language that is hosted on a web server and accessed via a browser-based IDE. This web-centric setup also helped students start coding in their browser without the frustrations of software installation, which prior work found to be a major barrier to getting started [29]: *"The goal was to minimize tool setup for students. Everything had to be done in the browser."* (P2)

In short, the CBDS team repurposed Chromebooks, which have traditionally been used for casual web browsing, as end-user programming tools for aspiring data scientists.

B. Repurposing Tools for Open and Reproducible Science

As academic researchers who practice open and reproducible science [51], [52], the CBDS team were adept users of computational notebooks – especially R Markdown [17] – to do end-user programming for their research. R Markdown allows users to write prose in the lightweight Markdown format, while interleaving graphs, diagrams, and runnable code in several programming languages such as R and Python. Users can write R Markdown in a text editor or in RStudio Cloud, which renders it as a notebook-like interface similar to Jupyter [53]. Since these are text documents, changes can easily be tracked in version control systems like Git. To create lessons for CBDS, the team repurposed this computational notebook – one of the central tools in their daily scientific workflow – to become the substrate for the educational content they built.

A related example of repurposing was the fact that years of the team's data analysis code and documentation were already in R Markdown, so they could be curated, simplified, and re-used for teaching. For instance, P9 took software documentation she had already written for internal lab use and adapted it into course content: *"The fact that all of the content is plain text makes making those changes super easy."*

Another example of repurposing was led by P1 and P3, who had both developed MOOCs before. MOOC providers like Coursera offer an in-browser rich-text editor where instructors are supposed to write lessons and assignments for their course. Compared to the team's usual open science workflows, which take advantage of the command line, Git, and other programmatic tools, the team felt hindered by the compulsory use of manually-driven web-based GUIs. P1 explained: *"The way the Coursera platform is set up, which isn't as simple as 'push to GitHub,' it makes [content updates] difficult."* Thus, to better integrate the end-user programming workflows they already used for scientific research into their desired teaching workflow, the team partnered with online publisher Leanpub [54] to develop a new course platform. Leanpub is currently a platform for taking Markdown-formatted documents and compiling them into eBooks. P1 and P3 already had experience publishing eBooks there, and they shared Leanpub's ethos of working with Markdown files. The team also valued Leanpub's pricing philosophy and applied it to CBDS: content on Leanpub follows a pay-what-you-want pricing model, which enables people to get it for free if desired.

The team worked directly with Leanpub, which built them a custom web application where they can upload R Markdown files and have them render as course webpages and assessments. This web app recognized custom Markdown syntax for elements such as multiple-choice questions and programming assignments with test cases. P3 appreciated how it was compatible with their existing text-based workflow: *"Leanpub catered very much to the idea of 'text to course.' Assessments as plain text was a very important feature."*

Using this custom platform, seven of nine team members (P1, P2, P4, P5, P6, P8, P9) developed technical course content solely in R Markdown files, tracked changes using Git, and collaborated on developing course modules (Table I) using GitHub across 25 different repositories. Course material development began in February 2018, and the first in-person cohort for CBDS started at the end of May 2018. The team credited the ability to repurpose their research workflows as critical for launching this initial version in just three months. Significant updating of materials continued as the first cohort made their way through the program, as changes were made based on student feedback.

Lastly, the CBDS team also had to grapple with teaching modern data science software libraries that were continuously updating and changing their APIs. When they previously used MOOC platforms like Coursera, whenever a library or API changed, they would need to spend hours navigating menus and GUIs to modify relevant course content that mentioned that library. This manual workflow was antithetical to the team's practice of reproducible research [52], [55], where all of the figures, tables, and reported statistics in a data analysis can be automatically re-compiled with one command whenever the underlying dataset is updated. Working with Markdown course materials allowed the team to use command-line tools to find and appropriately replace outdated content, similar to how they would update an outdated statistic or graph in light of updated input data while they were doing research.

Plain-text data formats, coupled with command-line tools and Leanpub's platform, enabled the CBDS team to *take an end-user programming approach to course development* instead of relying on manually-driven GUI-based content management portals typically used for online courses.

VII. End-User Programming to Build New Tools

Seven team members (P1–P6, P8) had previously developed bespoke R-language packages for performing domain-specific analysis tasks or for sharing algorithms and data from their published research studies. Besides repurposing the tools of computational science to programmatically generate online

course materials, these team members also engaged in end-user programming to build custom tools for themselves. Here we detail two such tools: Didactr for validating course content and Ari for expediting video production.

A. Didactr: Custom Software to Validate Course Content

The team developed various R packages to help them create, check, and deploy the data science lessons that went into CBDS. One of those packages, called Didactr, allowed team members to automatically validate lessons to make sure their Markdown was structured correctly before being uploaded to Leanpub. Lessons comprised two types of files: lecture videos that explained course concepts (see next section on Ari), and R Markdown files containing lesson readings, example code, and assessments to practice after each lesson.

Didactr parses these files using heuristics to make sure they are formatted to display properly on Leanpub's online course platform. Compiling the lessons and checking for errors locally with Didactr was faster and provided more useful error messages compared to uploading an error-laden lesson to Leanpub and manually checking on the web: *"Compiling courses on Leanpub takes time. Didactr allowed me to preempt errors that I would get on Leanpub so I could fix them locally and shorten the correctness-evaluation loop."* (P2) Ultimately Didactr served as a "command center" package which allowed the CBDS team to test and track the dozens of rapidly-evolving content files that constituted the course.

In the spirit of end-user programming for one's own needs, Didactr's features were built piecemeal in response to recurring bottlenecks the team faced while making course content. For instance, P3 worked closely with several other team members to better understand tasks they were initially doing manually: *"I asked the content creators, 'Show me what you do' and tried to then find APIs that would allow us to automate as much as possible."* And P2 recalled how closely she worked with teammates to extend Didactr on-the-fly: *"When I would tell [P3] there was a feature I wanted, he would sit with me and build it right in front of me."*

B. Ari: Custom Software for Text-Based Video Production

Other than writing lessons and assignments, the most time-consuming part for the CBDS team was recording and editing lecture videos. Videos in CBDS often feature an instructor giving a real-time demo of writing and running code or showing how to think through a data analysis task. If the API for a function in the featured analysis changes, then significant portions of the video must be re-recorded and re-edited. This problem is particularly pronounced in fields like data science, where industry-standard libraries are rapidly evolving. P1 and P3 remembered how costly it was to re-record videos for their past MOOCs whenever the code they were teaching had their APIs updated: *"With content that changes so often it's not feasible to reshoot videos, re-edit, et cetera, every time an API changes"* (P3).

To make it easier to create and update these code- and slide-based lecture videos, the CBDS team developed a custom R package called Ari that allowed them to automatically generate narrated lecture videos from the R Markdown documents they were already writing as part of their course materials. To use Ari, a creator first passes in a set of lecture slides and an accompanying text narration script. Ari uses Amazon's text-to-speech web API [56] to synthesize a machine voice to speak out the script and FFmpeg [57] to stitch together the lecture slides and spoken audio into a final compiled video file with the proper timings. Lecture slides can be generated from R Markdown files, but often team members opted to use more traditional GUI-based presentation tools like Google Slides.

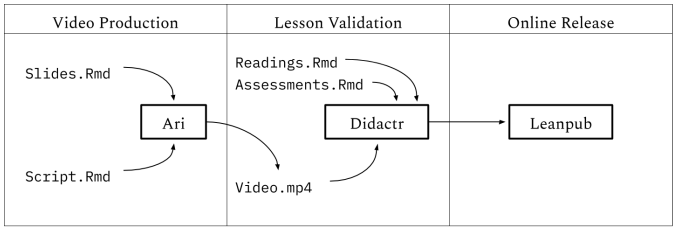

Fig. 1. The software workflow that the CBDS team developed to produce, validate, and release course content from R Markdown (`*.Rmd`) source files.

Ari helped the CBDS team take an end-user programming approach to video production, turning it into a process of editing text and compiling it into videos with R scripts instead of manually recording and editing using heavyweight video production GUI software. To make bug fixes or updates to their videos, they can simply edit text files and recompile. This format also allows them to easily track video edits in Git version control. P3 discussed how Ari's workflow aligned with the team's research philosophy: *"The videos are fully reproducible [from text-based sources], just like our scientific work."* P3 elaborated that having this more modular format meant they could iterate more quickly: *"This allows us to only modify content without changing presentation, delivery, et cetera. So we can take a much more experimental approach to making lecture videos."*

VIII. Toward End-User Software Engineering

Figure 1 shows the team's current course production pipeline where textual R Markdown files (`*.Rmd` extension) get compiled into videos (`*.mp4`), lesson webpages, and assessments, validated with Didactr, and then released online to Leanpub's web platform.

The CBDS team's goal was to create a free data science course for members of their local community, *not* to create a production-scale course development platform. However, to create such a course as a volunteer effort on a short time frame (3 months from inception to launch), they had to repurpose end-user programming skills from their research careers to create tools to make themselves more productive. But now that the course is in progress, the team found themselves transitioning from end-user programming into *end-user software engineering* [16], [18] with issues of tool maintenance, robustness, and updates from new contributors on their minds, especially as some of the original team members depart.

For instance, P1 was concerned about the ease with which future CBDS team members could interact with both the tools and course content: *"I know the original builders of the program will graduate soon, so lots of knowledge about maintaining the program will leave. This fact informed all of the technology decisions."* As our study was being conducted P2 finished her postdoc and moved to a new institution, and P1 explained the extent to which her departure was already testing the robustness of their tools: *"We have already done lots of maintenance and restructuring and our system is working. Team members who have then left are still regularly fixing bugs, which shows how easy the material is to maintain."*

A related concern was the extent to which tools could enable future maintenance and expansion of course content. P1 said the motivation behind building tools in the first place was the question: *"How can we make [course] maintenance costs as asymptotically close to zero as possible?"* But now the tools themselves need to be maintained and updated as well.

IX. Discussion

We conclude by reflecting on our findings in light of implications for future end-user programming and computing diversity research, end-user programming for education and social good, and the paradox of scale and access to education.

A. Implications for Future Research

This case study presents only a single snapshot, but we believe it can open the doors to future research on the interplay between end-user programming and diversity in computing. For instance, there is at least an order of magnitude more end-user programmers than professional software developers [58], [59], and they likely come from more diverse demographics than those who specialized in computing fields. Thus, one of the most practical and scalable ways to further broaden diversity in computing is to channel the energy of end-user programmers. How can institutions that employ such programmers foster these kinds of initiatives without making them too bureaucratic and thus undermining their bottom-up grassroots spirit? Can lessons from these volunteer-run efforts inspire new practices for designing collaboratively-constructed educational experiences?

Switching gears, how can systems researchers develop tools to better support the extensibility of end-user programming environments to stretch far beyond their original intended uses? In our case study, the CBDS team repurposed the R language ecosystem, which was originally designed for statistical research, to build an online course development platform. While experts in educational technology could probably come up with a "better" toolchain, the fact is that this is the toolchain these research scientists already know well, so tools should meet them where they are. But must every ecosystem reinvent the same wheels in an ad-hoc non-reusable manner? Or are there more general principles for constructing modern software platforms that we can abstract out into language-agnostic tools that developers can plug into whether they are working in R or Python or JavaScript or even spreadsheet environments?

B. DevOps Patterns in End-User Programming for Education

Reflecting on our nine interviews in this case study, one recurring theme was how much technical infrastructure was involved behind the scenes to keep CBDS running. It reminded us of how the past decade saw the emergence of DevOps [60]–[62], a practice that combines software *development* with the *operations* required to deliver and maintain that software. In industry, DevOps engineers write custom code to monitor the lifecycle of software products (especially web applications) throughout development, deployment, testing, and release. In a similar vein, the CBDS team are not only producing educational content like faculty normally do, but they are also writing custom software to manage the lifecycle of that content. In essence, they are mirroring the patterns in DevOps while performing end-user programming for education.

Like DevOps engineers, the CBDS team has significant influence on the design of their program (developing content), the programming tasks involved in delivering their "product" (building software to support that content), and monitoring its status (interacting with students to see where they get confused). Every CBDS team member can both make observations about what parts of their system (Figure 1) need to be improved and are empowered to make those improvements.

Also like DevOps engineers, the CBDS team repurposed or built custom software tools for each stage of the course lifecycle. They used the same tools that they would normally use to do their research to create course modules, deployed those modules on GitHub so other team members could collaboratively build upon them, developed their own monitoring software to test whether modules were formatted correctly, and released online and iterated based on student feedback. If not for their knowledge of appropriate tools to cobble together, they would likely not have been able to deliver CBDS on top of their normal duties as researchers. That said, some team members like P3 had concerns with the technical challenges of continued maintenance and scaling, given their multifaceted job roles: *"How can we be expected to be scientists, security experts, system administrators, and good instructors?"*

More broadly, we believe that treating educational artifacts like software artifacts by borrowing patterns from fields like DevOps could become a promising strategy as the demand for computing education grows in the coming years.

C. End-User Programming for Social Good

Another unique aspect of the CBDS project was how end-user programming was applied for broader social good. This project originated from the team's desire to create data science education opportunities for an underserved adult population that would not otherwise encounter an on-ramp into computing careers. Although the CBDS team was composed mostly of academic data scientists, it was not their data- or research-related skills that allowed them to build CBDS; rather it was their ability to design a code-based workflow that enabled rapid collaborative iteration on their course materials via end-user programming, software engineering, and DevOps.

The speed and relatively low cost with which CBDS was launched opens up the question: Who is in the best position to create such opportunities for underrepresented minorities to enter computing fields? Many existing diversity efforts have their origins at the top levels of organizations, whether it is CEOs diversifying hiring practices or nonprofits offering scholarships. This top-down approach, though impactful, is often not closely connected to the communities which these opportunities are designed for. Conversely, there are thousands of bottom-up local organizations working to help historically disenfranchised communities on the ground, but they are often not aware of paths into viable computing careers, especially in newer professions like data science. CBDS took more of a bottom-up approach: The team was not highly-positioned within the organizational structure of their university, and they relied on a partnership with the HEBCAC community organization to recruit interested students from the local area.

This case study points to the compatibility between a grassroots vision for positive social change and the spirit of end-user programming. CBDS was developed such that all team members could contribute to not just course materials but also to the custom software tools they developed for their own workflow. The flat structure and transparency within the team meant that they could address their students' concerns more quickly. Although the success of the program has yet to be determined, it may be good for top-down decision makers to rethink who should be empowered to help foster greater diversity in computing fields. Simply using free software or putting free course materials online is not enough to create lasting change in terms of who has access to computing education. It appears there was no single tool or innovation by the CBDS team that allowed this program to come to fruition. However, this program could not have been built without end-user programming, which equipped each team member with the level of technical agility required to respond to the needs of their student population.

We believe that the CBDS team's ability to write bespoke software to manage their course production pipeline, combined with their proximity to the target student population in their neighborhood, made them well-positioned to deliver such an educational program. In contrast, a traditional university instructor would likely not be able to produce and support a complex technical course outside their normal teaching duties and would also not have funding to hire professional engineering staff to help them. On the other end, a MOOC provider such as Coursera or Udacity would likely not create such a "small" program due to lack of perceived market size and revenue potential; to our knowledge, no major MOOC provider has yet partnered with local community organizations to produce courses for underserved adult populations.

D. Rethinking Scale and Access to Educational Opportunities

One critique of CBDS might be, *"How will this ever scale?"* At present, it does not scale, since the CBDS team must staff the online Slack chat channels and in-person office hours themselves on top of their day jobs as researchers. The team has ideas for how to gradually scale, such as using alumni as volunteer tutors for subsequent cohorts and fundraising to buy more Chromebooks. However, we believe the fact that the team did *not* initially think about scale was what led to this program being developed in the first place. If they had thought about scale from the beginning, they would have likely created an ordinary MOOC like what P1 and P3 have done before.

People who take MOOCs tend to already have higher incomes and higher levels of prior education; this is especially the case for courses that teach technical subjects such as computer science or data science [46]–[48]. It appears that if a course is designed upfront for reaching the largest possible audience in terms of enrollment, then it does not usually reach populations that have been historically excluded from educational opportunities. Thus, paradoxically, designing courses for scale might mean *less access* for those who are unlikely to find those resources on their own.

In contrast, CBDS presents an alternative to online courses or university outreach programs. It takes an approach where free course materials are designed to scale online but are also formatted in a way so that course creators can iterate on them quickly. But it was the team's willingness *to do what does not scale* – adapting to their students by partnering with the HEBCAC community organization – that enabled them to tailor the program continuously as new needs arose throughout deployment. Working with HEBCAC and the students face-to-face does not scale, but we believe this approach provides a path for greater access to computing opportunities.

Lastly, CBDS points the way toward future hybrids of online and in-person education. One idea here for scaling is that paid versions of CBDS could partially fund in-person versions that target historically underrepresented groups. With this financial model, those who have had more access to educational opportunities can pay to enroll and thus indirectly fund those who have not had access to the same opportunities. Beyond financial sustainability, online courses can take advantage of their ability to scale by transforming their online communities into in-person communities of support: Online alumni could be recruited to help with in-person tutoring in underserved communities and also provide guidance and networking opportunities related to computing careers.

X. Conclusion

We presented a case study of how a team of academic scientists repurposed end-user programming skills and tools from their research to create an adult education program to cultivate diversity in computing. The team provided an easily-accessible learning environment with free Chromebook laptops, a web-based coding platform, and weekly in-person office hours and online help. They customized R Markdown computational notebooks to develop and publish course content. And they built custom tools such as those to validate lessons and to compile textual scripts into lecture videos. This study shows how the bottom-up grassroots spirit of end-user programming can advance social good. Hopefully both small teams and large organizations can repurpose these lessons for positive ends.

ACKNOWLEDGMENTS

Thanks to the UC San Diego Design Lab for feedback, Xiong Zhang for feedback and BibTeX wizardry, and NSF award #1735234 for funding.

REFERENCES

[1] J. Margolis, *Stuck in the shallow end: Education, race, and computing*. MIT Press, 2010.

[2] S. U. Noble, *Algorithms of oppression: How search engines reinforce racism*. NYU Press, 2018.

[3] I. Bose and R. K. Mahapatra, "Business data mining – a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211 – 225, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037872060100091X

[4] G. Mann and C. O'Neil, "Hiring algorithms are not neutral," Harvard Business Review, December 2016.

[5] C. Lecher, "What happens when an algorithm cuts your health care," *The Verge*, 2018.

[6] V. Eubanks, *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press, 2018.

[7] C. Kelleher, R. Pausch, and S. Kiesler, "Storytelling alice motivates middle school girls to learn computer programming," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 1455–1464. [Online]. Available: http://doi.acm.org/10.1145/1240624.1240844

[8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592779

[9] D. Wolber, H. Abelson, and M. Friedman, "Democratizing computing with app inventor," *GetMobile: Mobile Comp. and Comm.*, vol. 18, no. 4, pp. 53–58, Jan. 2015. [Online]. Available: http://doi.acm.org/10.1145/2721914.2721935

[10] "Code2040: Join the largest racial equity community in tech," http://www.code2040.org/, accessed: 2019-05-01.

[11] "Girls who code: Join the girls who code movement!" https://girlswhocode.com/, accessed: 2019-05-01.

[12] "Black girls code: Imagine. build. create." http://www.blackgirlscode.com/, accessed: 2019-05-01.

[13] M. Guzdial, "Exploring hypotheses about media computation," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ser. ICER '13. New York, NY, USA: ACM, 2013, pp. 19–26. [Online]. Available: http://doi.acm.org/10.1145/2493394.2493397

[14] A. E. Tew, W. M. McCracken, and M. Guzdial, "Impact of alternative introductory courses on programming concept understanding," in *Proceedings of the First International Workshop on Computing Education Research*, ser. ICER '05. New York, NY, USA: ACM, 2005, pp. 25–35. [Online]. Available: http://doi.acm.org/10.1145/1089786.1089789

[15] M. Furst, C. Isbell, and M. Guzdial, "Threads™: How to restructure a computer science curriculum for a flat world," in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '07. New York, NY, USA: ACM, 2007, pp. 420–424. [Online]. Available: http://doi.acm.org/10.1145/1227310.1227456

[16] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 21:1–21:44, Apr. 2011. [Online]. Available: http://doi.acm.org/10.1145/1922649.1922658

[17] "R Markdown: Analyze. share. reproduce." https://rmarkdown.rstudio.com/, accessed: 2019-05-01.

[18] M. M. Burnett and B. A. Myers, "Future of end-user software engineering: Beyond the silos," in *Proceedings of the on Future of Software Engineering*, ser. FOSE 2014. New York, NY, USA: ACM, 2014, pp. 201–211. [Online]. Available: http://doi.acm.org/10.1145/2593882.2593896

[19] B. A. Nardi, *A small matter of programming: perspectives on end user computing*. MIT press, 1993.

[20] J. C. Carver, R. P. Kendall, S. E. Squires, and D. E. Post, "Software development environments for scientific and engineering software: A series of case studies," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 550–559. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.77

[21] J. Segal, "Some problems of professional end user developers," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, ser. VLHCC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 111–118. [Online]. Available: http://dx.doi.org/10.1109/VLHCC.2007.50

[22] P. Prabhu, T. B. Jablin, A. Raman, Y. Zhang, J. Huang, H. Kim, N. P. Johnson, F. Liu, S. Ghosh, S. Beard, T. Oh, M. Zoufaly, D. Walker, and D. I. August, "A survey of the practice of computational science," in *State of the Practice Reports*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 19:1–19:12. [Online]. Available: http://doi.acm.org/10.1145/2063348.2063374

[23] P. J. Guo, "Software tools to facilitate research programming," Ph.D. dissertation, Stanford University, May 2012.

[24] M. B. Kery, M. Radensky, M. Arya, B. E. John, and B. A. Myers, "The story in the notebook: Exploratory data science using a literate programming tool," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 174:1–174:11. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173748

[25] M. B. Kery and B. A. Myers, "Exploring exploratory programming," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2017, pp. 25–29.

[26] A. Rule, A. Tabard, and J. D. Hollan, "Exploration and explanation in computational notebooks," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 32:1–32:12. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173606

[27] G. Wilson, "Software carpentry: Getting scientists to write better code by making them more productive," *Computing in Science Engineering*, vol. 8, no. 6, pp. 66–69, Nov 2006.

[28] "Data carpentry: Building communities teaching universal data literacy," https://datacarpentry.org/, 2018, accessed: 2018-09-20.

[29] S. Kross and P. J. Guo, "Practitioners teaching data science in industry and academia: Expectations, workflows, and challenges," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3290605.3300493

[30] "VL/HCC 2019: Call for papers," https://human-se.github.io/vlhcc2019/call-for-papers/, accessed: 2019-05-01.

[31] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, "Programming by choice: Urban youth learning programming with scratch," in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '08. New York, NY, USA: ACM, 2008, pp. 367–371. [Online]. Available: http://doi.acm.org/10.1145/1352135.1352260

[32] B. DiSalvo, M. Guzdial, C. Meadows, K. Perry, T. McKlin, and A. Bruckman, "Workifying games: Successfully engaging african american gamers with computer science," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 317–322. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445292

[33] B. DiSalvo, S. Yardi, M. Guzdial, T. McKlin, C. Meadows, K. Perry, and A. Bruckman, "African american men constructing computing identity," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2967–2970. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979381

[34] B. DiSalvo, M. Guzdial, A. Bruckman, and T. McKlin, "Saving face while geeking out: Video game testing as a justification for learning computer science," *Journal of the Learning Sciences*, vol. 23, no. 3, pp. 272–315, 2014. [Online]. Available: https://doi.org/10.1080/10508406.2014.893434

[35] B. Ericson, S. Engelman, T. McKlin, and J. Taylor, "Project rise up 4 cs: Increasing the number of black students who pass advanced placement cs a," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '14. New York, NY, USA: ACM, 2014, pp. 439–444. [Online]. Available: http://doi.acm.org/10.1145/2538862.2538937

[36] B. Brinkman and A. Diekman, "Applying the communal goal congruity perspective to enhance diversity and inclusion in undergraduate computing degrees," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: ACM, 2016, pp. 102–107. [Online]. Available: http://doi.acm.org/10.1145/2839509.2844562

[37] L. Porter, M. Guzdial, C. McDowell, and B. Simon, "Success in introductory programming: What works?" *Commun. ACM*, vol. 56, no. 8, pp. 34–36, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2492007.2492020

[38] L. Porter and B. Simon, "Retaining nearly one-third more majors with a trio of instructional best practices in cs1," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 165–170. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445248

[39] S. A. Rebelsky, J. Davis, and J. Weinman, "Building knowledge and confidence with mediascripting: A successful interdisciplinary approach to cs1," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 483–488. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445342

[40] K. Benda, A. Bruckman, and M. Guzdial, "When life and learning do not fit: Challenges of workload and communication in introductory computer science online," *Trans. Comput. Educ.*, vol. 12, no. 4, pp. 15:1–15:38, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2382564.2382567

[41] P. J. Guo, "Older adults learning computer programming: Motivations, frustrations, and design opportunities," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 7070–7083. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025945

[42] B. Dorn and M. Guzdial, "Graphic designers who program as informal computer science learners," in *Proceedings of the Second International Workshop on Computing Education Research*, ser. ICER '06. New York, NY, USA: ACM, 2006, pp. 127–134. [Online]. Available: http://doi.acm.org/10.1145/1151588.1151608

[43] ——, "Learning on the job: Characterizing the programming knowledge and learning strategies of web designers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 703–712. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753430

[44] P. K. Chilana, R. Singh, and P. J. Guo, "Understanding conversational programmers: A perspective from the software industry," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 1462–1472. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858323

[45] J. M. Corbin and A. L. Strauss, *Basics of qualitative research: techniques and procedures for developing grounded theory*. SAGE Publications, Inc., 2008.

[46] L. Breslow, D. E. Pritchard, J. DeBoer, G. S. Stump, A. D. Ho, and D. T. Seaton, "Studying learning in the worldwide classroom: Research into edX's first MOOC," *Research & Practice in Assessment*, vol. 8, 2013.

[47] J. D. Hansen and J. Reich, "Democratizing education? examining access and usage patterns in massive open online courses," *Science*, vol. 350, no. 6265, pp. 1245–1248, 2015. [Online]. Available: https://science.sciencemag.org/content/350/6265/1245

[48] M. Guzdial, "Limitations of moocs for computing education- addressing our needs: Moocs and technology to advance learning and learning research (ubiquity symposium)," *Ubiquity*, vol. 2014, no. July, pp. 1:1–1:9, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2591683

[49] "Historic East Baltimore Community Action Coalition (HEBCAC): Thriving Baltimore Communities," https://hebcac.org/, accessed: 2019-05-01.

[50] "Welcome to RStudio Cloud: Do, share, teach and learn data science with R." https://rstudio.cloud/, accessed: 2019-05-01.

[51] J. S. S. Lowndes, B. D. Best, C. Scarborough, J. C. Afflerbach, M. R. Frazier, C. C. OHara, N. Jiang, and B. S. Halpern, "Our path to better science in less time using open data science tools," *Nature ecology & evolution*, vol. 1, no. 6, p. 0160, 2017.

[52] R. D. Peng, "Reproducible research in computational science," *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011.

[53] "Project jupyter," http://jupyter.org/, 2017.

[54] "Leanpub: publish early, publish often," https://leanpub.com/, accessed: 2019-05-01.

[55] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, and M. Taufer, "Enhancing reproducibility for computational methods," *Science*, vol. 354, no. 6317, pp. 1240–1241, 2016.

[56] "Amazon polly: Turn text into lifelike speech using deep learning," https://aws.amazon.com/polly/, accessed: 2019-05-01.

[57] "Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video." https://ffmpeg.org/, 2017.

[58] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, ser. VLHCC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 207–214. [Online]. Available: http://dx.doi.org/10.1109/VLHCC.2005.34

[59] C. Scaffidi, "Counts and earnings of end-user developers – https://www.linkedin.com/pulse/counts-earnings-end-user-developers-chris-scaffidi/," Sep. 2017.

[60] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016.

[61] L. Zhu, L. Bass, and G. Champlin-Scharff, "Devops and its practices," *IEEE Software*, vol. 33, no. 3, pp. 32–34, May 2016.

[62] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, May 2016.