"It Broadens My Mind": Empowering People with Cognitive Disabilities through Computing Education

Varsha Koushik and Shaun K. Kane
Department of Computer Science
University of Colorado Boulder
Boulder, CO, USA
{varsha.koushik, shaun.kane}@colorado.edu

ABSTRACT

Computer science education is widely viewed as a path to empowerment for young people, potentially leading to higher education, careers, and development of computational thinking skills. However, few resources exist for people with cognitive disabilities to learn computer science. In this paper, we document our observations of a successful program in which young adults with cognitive disabilities are trained in computing concepts. Through field observations and interviews, we identify instructional strategies used by this group, accessibility challenges encountered by this group, and how instructors and students leverage peer learning to support technical education. Our findings lead to guidelines for developing tools and curricula to support young adults with cognitive disabilities in learning computer science.

CCS CONCEPTS

• Human-centered computing → Accessibility → Empirical studies in accessibility

KEYWORDS

Cognitive disability; computer science education; accessibility.

ACM Reference format:

Varsha Koushik and Shaun K. Kane 2019. "It Broadens My Mind": Empowering People with Cognitive Disabilities through Computing Education. In *2019 CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK.* ACM, NY, NY, USA. Paper 514, 12 pages. https://doi.org/10.1145/3290605.3300744

ACM ISBN 978-1-4503-5970-2/19/05...$15.00.

DOI: https://doi.org/10.1145/3290605.3300744

Figure 1. Members of Code Club work together on programming projects. One member works with an instructor to solve a programming problem.

1 INTRODUCTION

Computer science education is quickly becoming a core skill for young people around the world. Developing computer science skills can lead to opportunities in higher education and can lead to gainful employment. It is estimated that there will be over one million job openings in the field of computing by the year 2020 [6]. Computer science skills are also essential to work in STEM and in many non-STEM fields [25]. Aside from learning skills directly related to computing careers, learning computer science can also develop computational thinking skills, which can be useful throughout one's life [4].

In recent years, many groups have examined barriers to participation in computer science. Organizations such as AccessComputing [7] and AccessCSForAll [18] have focused on identifying and addressing barriers encountered by students with disabilities while studying computer science. People with disabilities represent up to 15% of the K-12 student population [20] and many may experience accessibility issues when learning computer science.

While much research has addressed accessibility issues in computer science for people with vision-related disabilities (e.g., [1, 16, 34]), relatively little research has explored cognitive accessibility issues in computer science. One

barrier to including students with diverse cognitive abilities is the lack of pedagogical resources tailored to this population [13]. This is a chicken-and-egg problem, as there exist relatively few examples of computer science courses or workshops that address this population, making it more difficult to identify successful strategies for including this population in computer science education activities.

In considering how to include individuals with cognitive disabilities in computer science education, several questions arise. First, how do we adapt curricular materials to work best for this population? Second, what developer tools, technologies, and project types may best support this population in learning computer science? Third, how can we structure computer science educational activities to best support these learners? Much of the recent interest in computer science education has focused on the attainment of jobs [26], while the design for user empowerment approach [19] has focused on empowering people with disabilities to build their own assistive technology. What learning outcomes are ideal for computer scientists with cognitive disabilities?

To explore these issues, we present a qualitative study of a technology and programming group that targets young adults with moderate to severe cognitive disabilities. For two years, this group, which we will refer to as Code Club, has trained young adults with cognitive disabilities to work as information technology professionals, and has increasingly incorporated aspects of computer science into its curriculum. We report on a series of field observations and interviews with instructional staff and members of Code Club. Our research to date has focused on understanding how this group has developed its own computing curriculum, how they have identified and overcome accessibility barriers, and how members have gained new skills through their participation. By highlighting a program that has successfully reached this underserved population, we identify accessibility challenges and strategies for overcoming these challenges to create an inclusive computer science curriculum. Our research explores the following research questions:

RQ1. What accessibility challenges do members of this community encounter when learning computer science?
RQ2. What workarounds have they developed to address these challenges?
RQ3. How do members of the community work together to address individual and group accessibility challenges?
RQ4. How do members of the community believe they benefit from learning computer science?

2 RELATED WORK

2.1 Education and Cognitive Disabilities

Lewis [22] provides an overview of human-computer interaction challenges for people with cognitive disabilities. Lewis highlights challenges related to communication and working with complex written materials, and additionally notes that a major challenge is that individuals with cognitive disabilities are underestimated and are thus excluded from educational opportunities.

Much research about educational approaches for students with cognitive disabilities has built upon the Universal Design for Learning (UDL) framework [32]. UDL has been used to support computer science learning [14,15,28]. This approach emphasizes practices such as representing information in multiple formats, providing clear step-by-step instructions, interleaving instruction with inquiry activities, and facilitating peer-to-peer learning [23,33]. While the teaching staff at our field site did not formally follow UDL practices, their work provides an example of how these principles may be adopted in an informal educational setting.

2.2 Inclusive Computer Science Education

Much research on inclusive CS education has focused on making CS accessible to blind and visually impaired students. A blind student who is able to overcome inaccessible development tools can often perform as well as any sighted person and may pursue higher education in CS or a career in computing. Thus, research on the accessibility of CS education for blind users can benefit from studies of successful blind programmers (*e.g.,* [1]) and a comparatively large population of students who wish to learn CS. As a result, researchers have developed and tested a variety of alternative computer science tools for blind programmers, including alternative code editors [17,24], new programming languages [34], and tangible programming toolkits [37].

In comparison to blindness, very little research has explored challenges faced by people with cognitive disabilities or approaches to addressing these challenges. Much of the existing research focuses largely on people with mild learning or cognitive disabilities. For example, Powell et al. [27] explored how dyslexia affects the ability to learn computer science, and Thompson [38] studied the programming practices of children with dyslexia.

Almost no research has explored new programming tools for people with cognitive disabilities. Instead, much of this work has focused on pedagogical practices such as UDL,

and the use of existing programming tools. Much of this research has focused on block-based programming languages, which offer a simplified, highly visual environment for experimenting with code [29]. Block-based languages are often designed for children, although they may be useful to the general population as well, including adults with cognitive disabilities. Taylor et al. [31] introduced block-based programming to elementary school students with Down Syndrome and found that the students responded positively to the system's multimodal input and output capabilities. Another recent article [36] documented how one special education teacher has used block-based programming in their classroom. While our present study focuses on educational practices using existing technology, it provides insight into how these technologies may be adapted to support learners with cognitive disabilities.

2.3 Computer Science and Empowerment

There are many potential benefits to learning about computer science, and some of these benefits are especially important for people with disabilities. As noted previously, computer science skills may lead to employment opportunities. Buehler et al. [5] explored how people with cognitive disabilities could learn 3D printing skills and leverage these skills to create and sell objects, and recently Microsoft has begun a program to hire neuro-diverse engineers [30]. Developing computational thinking practices may lead to general improvements in problem solving skills [40]. Learning how to work with data and online media can also support self-expression and social connection [10, 21]. Finally, developing technical skills can help empower people to solve their own accessibility problems [12, 19]. This research is motivated by the belief that learning computer science can have many potential benefits.

3 FIELD STUDY

We conducted a series of observations, interviews, and demo sessions with instructors and students at Code Club (a pseudonym), a computer science educational program for adults with cognitive disabilities.

3.1 Field Site: Code Club

Code Club is an educational program within a larger day program that provides employment and independent living assistance to people with cognitive disabilities. Code Club has two sites, both in the United States. Code Club meets twice per week, for four hours per day, at each of the two sites. Both sites are managed by a single instructor, who we refer to as Sally. To the best of our knowledge, Code Club is the only day program for adults with cognitive disabilities that includes a computer science course.

Although Code Club and its associated community program do not provide explicit inclusion criteria, they describe their client population as "living with developmental disabilities, autism spectrum disorder, brain injury, mental illness, and often, accompanying physical challenges."

We began this research after meeting Sally at a conference about technology and cognitive disabilities. We discussed possible research opportunities for approximately six months, eventually agreeing on a research plan. The first author visited both of the Code Club sites, observing classes and conducting interviews with members and staff.

3.2 Participants

We interviewed two instructional staff and ten Code Club members. The members of Code Club are (mostly) young adults with cognitive disabilities, ranging in age from 20 to 50. We did not collect individual diagnoses from our participants as we did not believe this personal information was relevant to our research goals. However, all members had been admitted to Code Club (and its parent organization) due to one or more of the following diagnoses: Alzheimer's, autism, brain injury, memory disorder, developmental disability, or learning disability. All members experienced challenges with independent living. The study participants are described in Table 1.

Membership in Code Club was determined in part by the participant's ability to speak, read, and write. Sally reported that all members except one, Teigen, were able to read independently. Teigen was able to participate in Code Club with the assistance of a staff member, who read curricular materials to Teigen and constructed programs with her input. Another member, Mark, had limited speaking ability due to dysarthric speech and was unable to type due to a mobility impairment.

Code Club has two membership tiers. Members begin as trainees and are promoted to mentor after completing some initial programming tasks. Mentors are expected to make themselves available to help trainees and to teach when Sally is unavailable.

3.3 Recruitment and Consent Process

We received approval from our university's institutional review board before contacting any of the Code Club members. Our initial contact with the Code Club members was through Sally.

Table 1. Our study participants include instructional staff, senior students, referred to as mentors, and junior students, referred to as trainees. All names are pseudonyms chosen by the authors (S=staff, M=mentor, T=trainee).

Name	Age	Gender	Role	Reading/writing	Site 1	Site 2
Sally	53	Female	Staff	✓	✓	✓
Samantha	59	Female	Staff	✓	✓	
Monty	24	Male	Mentor	✓	✓	
Mark	31	Male	Mentor	Can read, but cannot type	✓	
Martin	23	Male	Mentor	✓		✓
Micah	27	Male	Mentor	✓		✓
Thomas	22	Male	Trainee	✓	✓	
Tim	25	Male	Trainee	✓	✓	
Tony	27	Male	Trainee	✓		✓
Tina	22	Female	Trainee	✓	✓	
Tiffany	41	Female	Trainee	✓		✓
Teigen	38	Female	Trainee	Needs assistance		✓

She introduced the program to the members, provided consent and assent forms, and returned the consent and assent forms to the research team. Because Sally's experience was crucial to understanding how Code Club works, we considered her a participant in our study; she had no access to participant data and had no official role on the research team beyond distributing recruitment information and consent forms.

Although all participants were over the age of 18, some participants were not their own legal guardians due to their disability. We provided consent forms for all participants who were their own legal guardians. All student members in Code Club completed an assent form, and their guardians completed a corresponding consent form.

Our consent forms requested the ability to observe, audio record, and take notes during class sessions; to interview participants; and to collect photographs to document the class. All students and staff members indicated their assent or consent to participate in the research. Participants were compensated for their time.

3.4 Data Collection

Our research team collected data at each site over two weeks. Data collection activities consisted of observations, interviews, and project demos from Code Club members.

Observation of Code Club Sessions. Our research team attended two class sessions at each site, participating in a total of 16 hours of class time. Class sessions included lecture presentations from Sally, group discussions, and project time. The structure of the teaching sessions was similar at both sites, although each site featured a different set of participants and took place in a different classroom. The first author observed the class sessions, took notes and pictures, and video recorded some group discussions.

Interview with Program Director. Our research team conducted several interviews with Sally, the director and founder of Code Club. We conducted a formal, 90-minute interview with Sally and participated in several brief follow-up conversations. Discussion topics included the formation of Code Club, her criteria for recruiting members into Code Club, and her teaching strategies.

Interviews with Members. Our research team conducted one-on-one and small group interviews with each member. Interviews occurred either before or after class sessions. Sally took part in each interview in accordance with the parent organization's policies. During Mark's interview, Sally helped interpret his responses due to his dysarthric speech. Interviews ranged between 15 and 90 minutes long based on the participant's level of engagement and on scheduling constraints. Interviews were audio recorded with the participant's permission.

Curricular Materials and Project Demos. Sally shared her curricular materials, including programming tutorials, project documentation, and course policies, with our research team. We collected and scanned these materials and discussed them during our interview with Sally. All Code Club members demonstrated at least one of their coding projects for our research team. Demonstration sessions occurred along with the interview sessions. Demonstration sessions were audio and video recorded.

3.5 Data Analysis

We analyzed all of the data, including interviews, observations, and documents, as a single data set. We used open coding [34] to identify themes in the data. All themes were initially identified by the first author and revised collaboratively by the research team.

4 FINDINGS

We collected and analyzed data regarding the structure of Code Club, curriculum design strategies, accessibility challenges faced while learning computing, and Code Club's peer mentoring model.

4.1 Forming Code Club

We discussed the formation of this program with Sally, the program director and primary instructor. Before starting Code Club, Sally worked as a social worker and SQL database administrator in Code Club's parent organization.

Sally was then promoted to the role of Director of Information Technology. As Director of IT, Sally was responsible for researching and testing new assistive technologies that might be helpful to the program's members. As she worked with members to test these technologies, Sally began to recognize their expertise in using and evaluating assistive technologies. Sally decided to formalize the members' technical training in an educational program that became Code Club:

> ***Sally:*** *I thought who better to help with this work than the people I work with and provide services to? They are the best testers in the world. They are the ones that are going to use the technology, who better to test it than somebody who has insight on how it's going to work because they are going to be the users. They are testing it because they are ultimately going to use it, they have insight about it.*

Code Club began with a focus on using and configuring a variety of technologies, especially smart home and Internet-of-Things technologies, with the hope that these skills might lead to employment. The computer science curriculum began in Code Club's second year, as Sally felt that members had mastered the smart home technologies and were ready to further develop their skills:

> ***Sally:*** *The coding came as an idea because getting more into SmartThings … you can also program SmartThings. If we want to get them to the point to do that, we need to step back and begin teaching them programming.*

4.2 Recruiting Club Members

While Sally noted that she was eager to grow Code Club, she found that identifying and recruiting new members was one of her most challenging tasks. Although some individuals sought out membership in Code Club, Sally usually found potential members by visiting group homes and participating in community events.

The official criteria for joining Code Club are: interest in technology, ability to work in a group, motivation, professional behavior, and an ability to commit to between six and twelve months of instruction. When asked to describe her personal criteria for recruiting members, Sally noted that members should have experience using technology in their homes, should have an interest in helping others, and should be able to independently read large blocks of text. However, some members were able to enter Code Club without meeting all of these criteria. For example, Teigen is unable to read independently, but was selected for her interest in the program, and she participates with the assistance of a staff member. Overall, Sally noted that most people who had entered Code Club were successful, although some early members had left the program because they found it stressful.

4.3 Curriculum Design Strategies

A major part of Sally's work involves choosing topics, finding appropriate teaching materials, and adapting those materials for use in Code Club. Because her background is primarily in social work, rather than in computer science, Sally often needs to teach herself how to use the technology before she can figure out how to teach it to Code Club.

Choosing Technology. The first step in creating Code Club's curriculum is to identify platforms and programming languages. Sally generally chooses technologies that she thinks students would be excited to learn about and that are at an appropriate level of difficulty. When choosing a new technology, Sally also considers how learning about that technology could support members' future educational or employment goals:

> ***Sally:*** *All the technologies that we choose are based on the needs of the people we serve.*

Initially, the Code Club curriculum focused on physical computing and assistive technologies. Sally encourages Code Club members to work on assistive technology projects as she feels members may have special expertise as users of assistive technologies. An early project that was particularly successful was a smart pillow that can be used by a person with limited speech to communicate with caregivers at home. The pillow detected the number of taps on the top of the pillow and, based on the tap count, speaks out a recorded phrase. The project consists of the pillow, some sensors from another smart home device, and a Makey-Makey board (Figure 2).

Figure 2. Code Club members demonstrate a prototype smart pillow. Sensors on the pillow detect taps and play an audio message based on the number of taps.

After members demonstrated success in working with smart home technologies, Sally felt that the group might be able to handle more complex technical challenges, including

creating their own computer programs. Sally examined several programming tools that targeted novice programmers, including code.org, Python, and Scratch. Sally eventually decided to teach Scratch, as she felt that its simple structure and visual nature could be appropriate for her members, and because she was able to find high-quality instructional resources online. Sally described Scratch as a tool that is appropriate for all ages:

> ***Sally:*** *The basis of Scratch is applicable to anybody … it's not just being used in school systems anymore. It's being used in … senior centers and high schools and everywhere else so the stigma that it is for kids is long gone.*

Choosing Course Materials. In addition to choosing a technology to teach, Sally noted that she spent a considerable amount of time searching online for curricular materials for her students. Sally described the criteria she used to identify lessons that would be appropriate for Code Club.

First, instructions must be simple and well-structured. Each step should be presented simply, with not too much text on screen, and with simple navigation between pages. Instructions with too much text or a complex page layout would be discarded. Sally often chose tutorials that were intended for K-12 students, as they often feature simple writing and clear lesson plans.

Second, Sally noted that instructions should use visual aids when possible. Many Code Club members enjoyed learning from video tutorials. Instruction pages should have straightforward visual layouts with clearly marked headings. These pages should also have example illustrations and checklists. Sally described one educational web site that she thought was too visually complicated for her students:

> ***Sally:*** *Like where do they start? They're going to be thrown off by the information on the left side, they're going to be thrown off by the information on the right side, they're lost. Anything that has … information that can throw you off, you're gone!*

Finally, the design of the programming language itself helps to guide the choice of lessons. For example, Scratch uses colors to distinguish different code elements, which is helpful for members who are less skilled with reading text.

At the time of our data collection, Code Club was following using a series of online Scratch[1] tutorials. Students began with a simple project that included heavily scaffolding. Scratch's remix feature proved useful here, as a student could easily build on an existing project. As a student developed their programming skills, they moved to more complex lessons with fewer explicit instructions. Finally, students moved to their own independent projects.

Members' projects typically started simply, but could grow to be quite sophisticated, including multiple types of audiovisual media along with computational concepts such as sequential program flow, conditional statements, loops, and input events.

4.4 Teaching Computer Science Thinking

In addition to learning how to write code, Code Club members practiced planning programs, debugging code, and seeking out help.

Learning How to Create a Program. Mentors and trainees often worked together to plan out programs before writing them. Members sometimes began by exploring the Scratch online repository for ideas, looking at other projects' code to see how the apps worked. This activity is easy in Scratch, as anyone can see the source code for any other project. Some members also wrote out plans for their program before writing their code. Sally encourages members to break down a project into smaller steps. For example, when member Monty decided to make a racing game, he first worked with Sally to write down the program structure on the whiteboard (Figure 3).

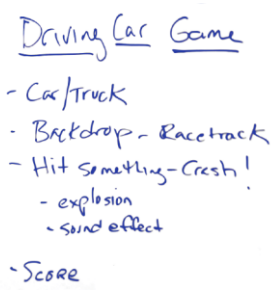

Figure 3. Plans for a driving game are sketched out on the whiteboard to help a student create the program.

Debugging and Getting Help. Code Club members used various strategies to identify and fix problems in their code. Most commonly, members would first ask another member for help. If the members were unable to solve the problem themselves, they might then ask Sally for help. Sally emphasized the importance of teaching the members to think through a problem, asking probing questions about the problem and encouraging them to break down the problem into smaller steps.

[1] scratched.gse.harvard.edu/guide

In some cases, Sally would ask a member to present their problem to the entire group. The member would then show their code on a large shared display and describe their problem to the group. This group conversation helped all members to learn to identify and solve common problems, while the member who was stuck often figured out what the problem was as they were explaining it to the group. This form of "rubber duck debugging" [11] was found to be helpful for many Code Club members.

4.5 Accessibility-Related Challenges

Members of Code Club often experienced typical programming problems during their work. For example, Monty demonstrated a Scratch animation in which two fish swam through an aquarium. Initially, one fish moved in the wrong direction. After talking through the problem with Sally and Mark, Monty realized that the fish sprite was accidentally rotated 90 degrees. In addition to these problems, Code Club members experienced some challenges that were less typical.

Reading and Understanding Code. Some Code Club members experienced difficulty in reading and understanding the structure of code blocks. Although Scratch's use of color to identify blocks was usually considered helpful, complex code structures with multiple blocks could still be difficult to understand.

While the color-coding of blocks was generally helpful, members sometimes became reliant on them, which could limit their ability to understand the code itself. Sally noticed this problem and began testing members by printing out lessons in black-and-white so that they would have to read the code rather than relying on the color of the blocks.

Gaps in Programming Knowledge. Code Club members typically followed a sequence of tutorial documents selected by Sally. In completing these tutorials, members practiced using language features such as sequential program flow, conditional statements, and loops. However, sometimes the lessons would skip or gloss over important concepts such as variables, and Code Club members tended to have little or no knowledge of how to use these features, causing them to get stuck when they attempted to create more complex programs. Because they were used to learning from Sally's hand-picked lessons, members were often unable to seek out help online, and instead required in-person help from Sally or a mentor.

Following Tutorials. Code Club members used both written and video tutorials when learning Scratch. Videos, GIFs and images could be especially useful for those members who were less confident readers. However, several members experienced a particular challenge when following along with image-heavy tutorials: they confused the tutorial window and the code editor window. For example, Figure 4 shows a Scratch code editor window with a tutorial open beside it. Members would occasionally confuse these two windows, accidentally clicking on images of code in the tutorial window rather than the "real" code blocks in their editor. Sally considered this a common problem:

> ***Sally:*** *Sometimes they don't realize you go all the way over to the left to click, and the on-screen tutorials are sometimes … too hard for somebody to do because they are trying to click on what they are being demonstrated on, so when it says create a new sprite and they show the picture of the new sprite, they are clicking on what it's showing them versus going to the left side to click it.*

To address this problem, Sally often provided a member's early lessons as a paper print-out, rather than pointing them toward an online tutorial.

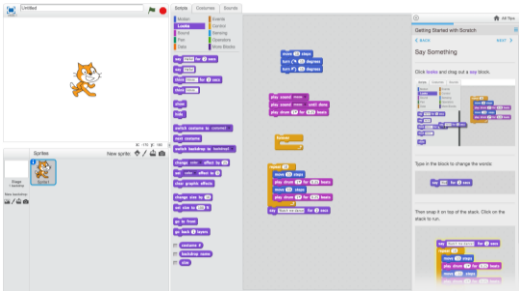

Figure 4. Scratch code editor next to a tutorial. Tutorial images were sometimes mistaken for the code editor itself.

Time and Project Management. Although Sally attempted to find projects that could be completed within the Code Club meeting schedule, members sometimes struggled to complete their projects. When members become tired or frustrated, they sometimes sit quietly instead of working, as they are not always able to leave the group meeting on their own. These problems could sometimes be addressed by helping a member through a difficult task, or by giving them a new project to work on.

Problems with Assistive Technology. Some Code Club members had other disabilities and required the use of assistive technologies. These technologies sometimes caused problems when programming. For example, Mark had recently begun using a head mouse to control his on-screen pointer. Initially, Mark struggled to use the head mouse and asked the staff several times to reconfigure its settings. After finally finding the appropriate configuration, the head mouse's battery died, leaving Mark unable to use the computer himself. Fortunately, staff member Samantha was available, so Mark was able to dictate his code to her,

although he was unable to complete the project independently as he had originally hoped.

Because Code Club computers were shared between members, one member's settings sometimes interfered with another member's work. In one session, Tiffany attempted to open Scratch but found that her mouse was configured such that the cursor moved in the opposite direction from the mouse. She expressed frustration about the situation but could not explain the problem and thus could not request help. After some time, Sally came over to see her screen and helped her correct the mouse settings.

4.6 Collaborative Work and Peer Mentoring

Although Code Club's peer mentoring model was initially developed in part due to the lack of instructional staff, it has become central to how members see the program. Code Club members work collaboratively in a number of ways.

Mentoring and Teaching. After completing some introductory programming tasks, members are promoted from trainee to mentor. Mentors are expected to help trainees with their technical problems; because mentors have completed the trainee phase, they often recognize the trainee's problems and know how to solve them. Mentors also run class sessions once per week while Sally is leading class at the other site.

Several members expressed pride in their role as mentor and teacher. During his interview, Monty shared a prepared statement about his role as a teacher:

> ***Monty:*** *It's that the more I get the people that I teach involved, the more they're willing to learn. The more I ask them questions and get them to understand, the more things they do on the computer gets them more involved.*

Sally considers this peer mentoring to be a core part of how Code Club members learn:

> ***Sally:*** *The more they [teach], the more they learn it. That's the way I learn. The way I've seen most people learn best, is if you can teach it, you know it. The more they're teaching it, the more they know it.*

Different mentors adopted different specialties. For example, Martin was especially interested in learning about different types of assistive technologies and took an active role in researching new technologies.

Helping People Like Themselves. Some mentors expressed that they found it rewarding to help other individuals with cognitive disabilities. When discussing his teaching work, Monty expressed pride in his ability to help others:

> ***Monty:*** *For me, it's a great experience to work with people who have the same disabilities as me, well, not the same but almost the same, and be able to teach them something that they have never done before.*

When working with trainees who had similar disabilities, some mentors noted that they felt that they had particular insights into the challenges the trainees experienced.

Sometimes members were able to work with each other when they had difficulty requesting help from an instructor. Martin and Micah, who are both infrequent communicators, typically worked alone and rarely asked for help. However, during one project, Martin became stuck and asked Micah for help. Micah offered his help, and the two worked together for the remainder of the project.

Dividing Labor. In some cases, a member was unable to complete a task, either due to a lack of understanding or due to an accessibility barrier. In these cases, group members would sometimes break down a task in order to solve a problem together. For example, when Mark's head mouse stopped working, Samantha was able to input his commands into the computer. In another case, Mark, Monty, and Sally were working together to build a physical computing project involving multiple sensors. Because of Mark's limited mobility, he focused on writing the program code using a tablet, while Monty placed the physical components and Sally connected the wiring. Although this project would have been a challenge for any one member, the group was able to complete the project by working together, each member choosing the appropriate task for their abilities in a form of collaborative accessibility [3].

Supporting the Community. To help Code Club members gain practical knowledge, Sally incorporated a public service model into the Code Club curriculum. In this program, members go to group homes or other community facilities and set up smart home technology. Members participate in several community technology sessions, gaining independence with each subsequent visit, first following instructions from Sally, then providing instructions to Sally, and finally working alone. Currently, most members are still in the first stage; only Monty and Mark have completed an independent project, which was setting up a Wi-Fi-powered, color-changing light bulb in a classroom of one of the day programs.

4.7 Outcomes Beyond the Classroom

Most of our conversations with Code Club members focused on their programming activities within the class. However, members also discussed how participating in Code Club has impacted their lives outside of class, and

how their computer science work helped to support their long-term goals.

Career Goals. One of Sally's primary goals in creating Code Club was to prepare members for jobs in which they could use their technical skills. She hopes that members will "learn … to be able to truly code and get jobs in the fields they choose." Members often expressed interest in technical careers. When asked about his career goals, Monty said:

> ***Monty:*** *I hope that one day I can have a job with something like this, that's kind of my intention, look forward to a job with technology and that kind of stuff … Honestly [my dream job] would be to work at Google.*

Since Code Club began, one member graduated from the program and accepted a job at the local library, where he works on the library's social media outreach activities.

Helping Others. Several members talked about how they had used their technical skills to solve problems in their everyday lives. Tina described how she helped her mother fix a problem with her mobile phone: the phone was turned off and her mother did not know how to turn it back on. Because of her experiences with technology, Tina knew to hold down the power button to turn it back on:

> ***Tina:*** *It wouldn't come on, just had to hold the power button and turn [it] on.*

Both Mark and Monty described how they were able to answer technical questions from friends and colleagues in their group home. Mark described helping friends with mobile applications and smart home devices. Monty mentioned that practicing his teaching skills in Code Club helped him teach staff members in his group home:

> ***Monty:*** *It's really interesting that I can teach [Code Club] and be able to teach my staff. If they don't know how to set up something, I can help.*

Developing Social Skills. Sally noted that an often unstated but important goal of Code Club is to promote social skills such as leadership, teamwork, timeliness, and self-confidence. Several members discussed how participating in Code Club helped them develop new skills. For example, Monty described how Code Club helped him to discover his fondness for teaching:

> ***Monty:*** *I love to teach and show my knowledge about technology and give it to other people.*

While describing what he has learned in Code Club, Monty also noted that learning to program led him to a new way of thinking:

> ***Monty:*** *Programming for me is kind of like … it broadens my mind a little bit. It kind of makes me smart, I don't know, a little smarter than I originally was. The more I learn, the more I can teach other people.*

Code Club has motivated members to feel more confident about their own abilities. When Martin first joined Code Club, he planned on finishing the course and going back to live with his parents. After participating in Code Club for a year, Martin is now committed to finding a job and becoming more independent.

Finally, participating in Code Club has also helped some members practice their social skills. When Tina first joined the group, she would cry if the instructors spoke to her. After four months in Code Club, Tina is now able to speak to her peers and agreed to participate in this research study. Micah, who was shy and noncommunicative when he began the program, now frequently helps his peers with their projects.

5 DISCUSSION

Our initial inquiry into Code Club was motivated by an interest in understanding whether it was successful and, if so, how it was able to be successful. To the best of our knowledge, Code Club is the only program of its type in North America. In attempting to understand Code Club's success, several factors stood out: the program director's combined expertise in social work and technology, the focus on developing technical and professional skills for a future career, and the comprehensive use of peer mentoring. While Code Club is a unique organization, these qualities could certainly be passed on to other programs, and we hope that our exploration of what makes Code Club work can lead to the development of similar programs in the future.

In studying Code Club and its members, we also sought to identify any unique accessibility challenges experienced by members of this community. We found that Code Club members experienced many of the challenges that anyone would experience while learning technology. When members did encounter an unusual challenge, such as the confusion between the code editor and tutorial document, the mentors and staff were often able to find a solution. Documenting these challenges and workarounds may lead to new tools and curricula to better support people with cognitive disabilities in learning computer science.

Finally, we sought to understand what types of incentives might best motivate adults with cognitive disabilities to learn computer science. As with many computer science

students, Code Club members were often motivated to develop their technical skills in order to seek a career in a computing-related field. However, we also found that members were motivated to develop new assistive technologies and to teach and support other people with disabilities. Members were also motivated by the belief that participating in their program could lead to improved social and professional skills, and several members reported experiencing real improvements in these areas. Understanding these motivations may support the development of educational tools that can help this population achieve their goals.

6 IMPLICATIONS FOR INCLUSIVE EDUCATION

Code Club serves as a successful example of engaging people with cognitive disabilities in computer science. Many Code Club practices reflect Universal Design for Learning (UDL) principles such as presenting information in multiple modalities, breaking down problems into discrete steps, and facilitating peer learning. However, Code Club's practices evolved through trial and error, and therefore may provide insights beyond the maxims of UDL. Here we provide an overview of successful strategies articulated by Code Club's educational team, as well as insights from our experience as researchers within this community.

6.1 Code Club's Stated Principles

While Code Club's educational practices are complex and are continually evolving, a few lessons repeatedly surfaced throughout our study:

Provide simple, well-structured, and modular activities. Activities should be presented as a series of clearly-defined steps so that the next action is always clear. Modular activities should support work that occurs over multiple sessions with minimal effort needed to jump back in. Adding checklists and question prompts between steps can help to keep learners on track and can make it easier for a teacher or peer to help when a student gets stuck.

Use carefully designed visual aids. Educational materials should use color and visual layout to convey information. Clear visual design should be applied across all learning materials, including written instructions, the code editor, and even the programming language itself. Provide clear differences between the appearances of software tools and tutorials or other documents that may depict those tools.

Support peer teaching and learning. Peer mentoring can reduce the burden on teaching staff, but also serves as a powerful motivator for students to advance through the program. Providing explicit stages of advancement from trainee to mentor may help reinforce the responsibilities of mentorship.

6.2 Insights from Our Research

Beyond the explicitly stated principles behind Code Club's educational program, we note these additional practices that have helped to support Code Club's success, and that may help to support similar programs in the future:

Anticipate multiple disabilities and assistive technologies. Code Club members used a diverse set of assistive technologies that sometimes interfered with the system software or with other members' assistive technologies. Test all educational materials with a representative set of assistive technologies. On shared devices, provide methods for easily changing the user profile.

Support teams with complementary abilities. Code Club members often enjoyed working together, and sometimes used groupwork to overcome an individual's accessibility challenges. Provide opportunities for learners of different abilities to work together. Design activities that can be broken down into different types of work, such as planning, writing code, and assembling hardware.

Encourage diverse goals and outcomes. Code Club members were motivated by a variety of goals: getting a job, helping their friends, giving back to their community, and participating in social activities. Within a diverse group of learners, not all goals may be achievable or desirable to everyone. Encourage students to articulate their goals and provide structures for tracking progress towards them.

7 FUTURE WORK

We are excited to continue our collaboration with Code Club, both to explore how to support the existing program and its members as well as to increase our understanding of how to create and support similar programs in the future.

One area of future research is to explore how members of this community can transfer from Scratch and Makey-Makey to mainstream programming languages. Supporting knowledge transfer between education-focused tools and mainstream programming languages presents a number of challenges, and these challenges will likely play out differently for Code Club members than for the general population. A related challenge is in identifying the technical skills developed by Code Club members and mapping them to possible career paths.

A second area of research is in developing software tools to overcome some of the accessibility challenges encountered by members of this community when programming. For example, we could design a software development environment that provide clear visual structure, supporting learners who have difficulty with extensive text or complex code structure. We could also explore how tutorials could provide clear, multimodal explanations and instructions.

A third area of research is to consider how these accessibility challenges vary between individuals, and to explore adaptive systems that can build a profile of a user's abilities and provide personalized support.

Finally, as peer mentoring is a central part of Code Club, we may explore technologies to support the mentoring process, such as by providing tools for peers to share code, debug each other's programs, or collaborate over a distance.

8 CONCLUSION

Studying computer science skills is seen by many as a way to empower individuals, providing them with a potential career path and supporting development in computational thinking and other areas. Despite the great interest in introducing young people to computer science, people with disabilities are still excluded from many of the benefits of computer science education. This exclusion is especially severe for people with cognitive disabilities, as few resources exist for including these individuals in computer science education. In this work, we have examined one organization that has successfully tackled many of these problems and shown that it is possible to adapt computer science curricula to support the goals and abilities of young people with cognitive disabilities. Understanding the challenges encountered by this group, and how they have been overcome, may lead to more inclusive approaches to teaching computer science.

ACKNOWLEDGMENTS

We thank our participants for taking part in our study. We also thank Erin Buehler, Jed Brubaker, Clayton Lewis, and Amy Voida for giving us valuable feedback. This work was supported by the National Science Foundation under grants IIS-1619384 and IIS-1652907. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Khaled Albusays and Stephanie Ludi. 2016. Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study. In Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '16), 82–85. DOI: https://doi.org/10.1145/2897586.2897616

[2] Jeffrey P. Bigham, Maxwell B. Aller, Jeremy T. Brudvik, Jessica O. Leung, Lindsay A. Yazzolino, and Richard E. Ladner. 2008. Inspiring blind high school students to pursue computer science with instant messaging chatbots. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08). ACM, New York, NY, USA, 449-453. DOI: https://doi.org/10.1145/1352135.1352287

[3] Stacy M. Branham and Shaun K. Kane. 2015. Collaborative Accessibility: How Blind and Sighted Companions Co-Create Accessible Home Spaces. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). ACM, New York, NY, USA, 2373-2382. DOI: https://doi.org/10.1145/2702123.2702511

[4] Karen Brennan, & Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (pp. 1-25).

[5] Erin Buehler, William Easley, Samantha McDonald, Niara Comrie, and Amy Hurst. 2015. Inclusion and Education: 3D Printing for Integrated Classrooms. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '15). ACM, New York, NY, USA, 281-290. DOI: https://doi.org/10.1145/2700648.2809844

[6] Bureau of Labor Statistics, U.S. Department of Labor, Occupational outlook handbook, Computer and Information Research Scientists. Retrieved April 16 2018 from https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm

[7] Sheryl Burgstahler and Richard Ladner. 2006. An alliance to increase the participation of individuals with disabilities in computing careers. SIGACCESS Access. Comput. 85 (June 2006), 3-9. DOI=http://dx.doi.org/10.1145/1166118.1166119

[8] Franklin, D., Hill, C., Dwyer, H., Hansen, A., Iveland, A., and Harlow, D. Initialization in Scratch: Seeking Knowledge Transfer, SIGCSE, 2016.

[9] Gerhard Fischer. 2011. Understanding, fostering, and supporting cultures of participation. interactions 18, 3 (May 2011), 42-53. DOI: https://doi.org/10.1145/1962438.1962450

[10] Andrea Forte and Mark Guzdial. 2004. Computers for Communication, Not Calculation: Media as a Motivation and Context for Learning. In Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4 (HICSS '04), Vol. 4. IEEE Computer Society, Washington, DC, USA, 40096.1-.

[11] Andrew Hunt, & David Thomas. 2000. The pragmatic programmer: from journeyman to master. Addison-Wesley Professional.

[12] Amy Hurst and Jasmine Tobias. 2011. Empowering individuals with do-it-yourself assistive technology. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11). ACM, New York, NY, USA, 11-18. DOI: https://doi.org/10.1145/2049536.2049541

[13] Maya Israel, Quentin M. Wherfel, Jamie Pearson, Saadeddine Shehab, & Tanya Tapia. 2015. Empowering K–12 students with disabilities to learn computational thinking and computer programming. TEACHING Exceptional Children, 48(1), 45-53.

[14] Maya Israel, Jaime N. Pearson, Tanya Tapia, Quentin M. Wherfel, & George Reese. 2015. Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. Computers & Education, 82, 263-279.

[15] Yasmin B. Kafai, Quinn Burke, & Mitchel Resnick. 2014. Connected code: Why children need to learn programming. Mit Press.

[16] Shaun K. Kane and Jeffrey P. Bigham. 2014. Tracking @stemxcomet: teaching programming to blind students via 3D printing, crisis

management, and twitter. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 247-252. DOI: http://dx.doi.org/10.1145/2538862.2538975

[17] Varsha Koushik and Clayton Lewis. 2016. An Accessible Blocks Language: Work in Progress. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '16), 317–318. DOI: https://doi.org/10.1145/2982142.2982150

[18] Richard E. Ladner and Andreas Stefik. 2017. AccessCSforall: making computer science accessible to K-12 students in the United States. SIGACCESS Access. Comput. 118 (July 2017), 3-8. DOI: https://doi.org/10.1145/3124144.3124145

[19] Richard Ladner. 2014. Design for user empowerment. In CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14). ACM, New York, NY, USA, 5-6. DOI: https://doi.org/10.1145/2559206.2580090

[20] Richard E. Ladner and Maya Israel. 2016. "For all" in "computer science for all". Commun. ACM 59, 9 (August 2016), 26-28. DOI: https://doi.org/10.1145/2971329

[21] Amanda Lazar, Raymundo Cornejo, Caroline Edasis, and Anne Marie Piper. 2016. Designing for the Third Hand: Empowering Older Adults with Cognitive Impairment through Creating and Sharing. In Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16). ACM, New York, NY, USA, 1047-1058. DOI: https://doi.org/10.1145/2901790.2901854

[22] Clayton Lewis. 2005. HCI for people with cognitive disabilities. SIGACCESS Access. Comput. 83 (September 2005), 12-17. DOI=http://dx.doi.org/10.1145/1102187.1102190

[23] Matthew T. Marino, Chad M. Gotch, Maya Israel, Eleazar Vasquez III, James D. Basham, & Kathleen Becht. 2014. UDL in the middle school science classroom: Can video games and alternative text heighten engagement and learning for students with learning disabilities?. Learning Disability Quarterly, 37(2), 87-99.

[24] Lauren R. Milne. 2017. Blocks4All: making block programming languages accessible for blind children. *ACM SIGACCESS Accessibility and Computing*, 117: 26–29.

[25] National Science Foundation. 2009. A week to focus on computer science education (Press Release 09-234).

[26] Hadi Partovi. 2014. Transforming US education with computer science. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 5-6. DOI: http://dx.doi.org/10.1145/2538862.2554793

[27] Norman Powell, David Moore, John Gray, Janet Finlay, & John Reaney. 2004. Dyslexia and learning computer programming.

[28] Meg J. Ray, Maya Israel, Chung eun Lee, and Virginie Do. 2018. A Cross-Case Analysis of Instructional Strategies to Support Participation of K-8 Students with Disabilities in CS for All. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). ACM, New York, NY, USA, 900-905. DOI: https://doi.org/10.1145/3159450.3159482

[29] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: programming for all. Commun. ACM 52, 11 (November 2009), 60-67. DOI: https://doi.org/10.1145/1592761.1592779

[30] Meredith Ringel Morris, Andrew Begel, and Ben Wiedermann. 2015. Understanding the Challenges Faced by Neurodiverse Software Engineering Employees: Towards a More Inclusive and Productive Technical Workforce. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '15). ACM, New York, NY, USA, 173-184. DOI: https://doi.org/10.1145/2700648.2809841

[31] Daniel Rezac. 2018. Coding for Special Ed? It's Real and It's Helping. Retrieved April 16 2018 from https://www.tynker.com/blog/articles/ideas-and-tips/coding-special-populations/

[32] David H. Rose, & Anne Meyer. 2002. Teaching every student in the digital age: Universal design for learning. Association for Supervision and Curriculum Development, 1703 N. Beauregard St., Alexandria, VA 22311-1714 (Product no. 101042: $22.95 ASCD members; $26.95 nonmembers).

[33] Snodgrass, Melinda. R., Israel, Maya., & Reese, George. C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers & Education, 100*, 1-17.

[34] Andreas Stefik and Susanna Siebert. 2013. An Empirical Investigation into Programming Language Syntax. Trans. Comput. Educ. 13, 4, Article 19 (November 2013), 40 pages. DOI=http://dx.doi.org/10.1145/2534973

[35] Anselm Strauss, & Juliet M. Corbin. 1990. Basics of qualitative research: Grounded theory procedures and techniques. Sage Publications, Inc.

[36] Matthew S. Taylor, Eleazar Vasquez, & Claire Donehower. 2017. Computer programming with early elementary students with Down syndrome. Journal of Special Education Technology, 32(3), 149-159.

[37] Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (DIS '17). ACM, New York, NY, USA, 739-752. DOI: https://doi.org/10.1145/3064663.3064689

[38] Rob Thompson. 2016. Teaching coding to learning-disabled children with Kokopelli's World. In Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on (pp. 258-259). IEEE.

[39] David Weintrop and Uri Wilensky. 2015. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. In Proceedings of the eleventh annual International Conference on International Computing Education Research (ICER '15). ACM, New York, NY, USA, 101-110.

[40] Jeannette M. Wing. 2006. Computational thinking. Commun. ACM 49, 3 (March 2006), 33-35. DOI: https://doi.org/10.1145/1118178.1118215