Neurodiverse Programmers and the Accessibility of Parsons Problems: An Exploratory Multiple-Case Study

Carl Haynes-Magyar
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
chaynesm@cs.cmu.edu

ABSTRACT

Parsons problems are computer programming puzzles that require learners to place code blocks in the correct order and sometimes indentation. Introductory computer programming instructors use them to teach novices how to code while optimizing problem-solving efficiency and cognitive load. While there is research on the design of Parsons problems for programmers without disabilities and programmers with visual or motor impairments, research regarding their accessibility for programmers with cognitive disabilities is scant. To identify the accessibility barriers and benefits of Parsons problems for neurodiverse programmers, an exploratory multiple-case study was conducted. Participants were asked to read eight chapters of an interactive eBook on Python and to solve Parsons problems. Within-case analyses of 15 retrospective think-aloud interviews with five novice programmers with disabilities led to four recommendations for improving the cognitive accessibility of Parsons problems. For example, programmers with seizure disorders may experience seizures when solving programming problems that require numeric calculations. Hence, creating a range of Parsons problems that do not require mental arithmetic could improve the learning experience for programmers with seizure disorders and those who struggle with mental calculations by lowering their cognitive load. Given this study's qualitative and exploratory approach, it does not offer conclusive, broadly generalizable results. Yet, it reveals detailed and promising avenues for exploration in computing education research that might elude many quantitative techniques.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in accessibility**.

KEYWORDS

Cognitive Accessibility, Inclusive Assessment, Introductory Programming, Neurodiversity, Parsons Problems

ACM Reference Format:
Carl Haynes-Magyar. 2024. Neurodiverse Programmers and the Accessibility of Parsons Problems: An Exploratory Multiple-Case Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630898

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630898

1 INTRODUCTION

While 3.48% of programmers are physically (visibly) disabled, 22.6% of programmers are neurodiverse (invisibly disabled) [44]. In particular, the percentage of developers who identify as having a concentration or memory disorder (10.6%), anxiety disorder (10.3%), and mood or emotional disorder (9.7%) has increased since 2021 [44, §Neurodiversity]. Learners with cognitive, learning, and neurological disabilities represent 33% of nearly seven million disabled students with the potential to be computer programmers [28], but several things may dissuade them from learning how to code. In particular, challenges to providing computing education to students with disabilities include choosing the right pedagogical approach and creating accessible learning materials and technologies [34, §Students with Disabilities].

Parsons problems are an active learning pedagogical technique that require learners to place blocks in the correct order and sometimes the correct indentation; they can also include distractor blocks that are not a part of the correct solution [16]. Prior research has provided evidence that Parsons problems and their variants can improve problem-solving efficiency, lower cognitive load, teach learners to identify and apply programming patterns, challenge learners, and be of use for learning to most undergraduate novice programmers [16, 19, 23, 24, 60]. Studies have also shown that distractors make these problems more difficult, increase self-reported extraneous cognitive load (i.e., the complexity of how the information to be learned is presented), and can increase time-on-task without increasing problem discrimination [14, 22, 53]. But are these drag-and-drop computer programming practice problems accessible to neurodiverse learners?

Neurodiversity signifies individual variation in cognitive function, behavioral traits, and affect; it is an umbrella term considered a 'moving target' that is useful for how it helps us imagine the world from something other than a neurotypical perspective—thereby decentering cognitive norms about, for example, rates of learning [5, 9, 52]. Most research on learning design has focused on neurotypical individuals [35], and there is relatively little computing education research on learners with cognitive disabilities and how they learn to program [12, 33]. In a recent systematic literature review on Parsons problems, researchers also pointed out the lack of investigations into the experiences of underrepresented programmers such as neurodiverse learners [16]. Hence, the research question addressed in this study was:

RQ1: What do neurodiverse learners report are accessibility barriers or benefits when solving Parsons problems?

This is the first research study focused on the accessibility of Parsons problems for novice programmers with cognitive, learning, and neurological disabilities (also referred to as neurodiverse). The design considerations highlight both the individual and the overall needs of programmers with disabilities that could have a curb-cut effect—that is, programmers without disabilities could benefit from addressing the needs of neurodiverse programmers. Each design recommendation is grounded in prior research within and outside the field of computing education. Importantly, this is an exploratory multiple-case study with a small *N* (see [38]) that was intended to be a careful qualitative analysis of neurodiverse programmers solving Parsons problems. Additional studies would need to be conducted in other contexts with programmers with and without disabilities to determine the generalizability of these findings.

2 RELATED WORK

This section reviews the literature on 1) neurodiverse learners and 2) Parsons problems.

2.1 Neurodiversity

Neurodiversity refers to individual variation in cognitive function, behavioral traits, affect, and sensory functioning differing from the general or 'neurotypical' population [49]. Judy Singer, a sociologist and autistic rights advocate, coined the term in 1999 [52]. Proponents view autism, attention deficit hyperactivity disorder, Tourette syndrome, dyslexia, hearing voices, bipolar disorder, down syndrome, dementia, and other neurominority experiences "as components on a broader continuum of sensory, affectual, and cognitive processing" [49, p. 2]. Neurodiversity is a challenge to the deficit (medical) model that portrays neurominorities as "ill, broken, and in need of fixing" where neurological deficits/disorders are exclusive to the individual [48, p. 1]. In contrast, the social model of disability concerns external forces that enforce restrictions on disabled people [49].

Most research on learning design has focused on neurotypical individuals [35], and there is relatively little research in computing education on learners with cognitive disabilities [33]. Accessibility research has disproportionately focused on blind and low-vision users [36]. Autism, intellectual or developmental disabilities (IDD), and cognitive impairment account for under 10% of papers within accessibility research and case studies only account for 4.0% of methods used [36]. Hence, this paper presents an exploratory multiple-case study of five participants with distinct and slightly overlapping cognitive disabilities to fill the gaps. This study carefully analyzes neurodiverse programmers' engagement with an interactive Python eBook that includes Parsons problems.

2.2 Parsons Problems

Parsons problems are drag-and-drop practice exercises that require learners to place code blocks into the correct order and sometimes indentation. In 2006, Dale Parsons and Patricia Haden developed them for introductory programming courses to maximize engagement, help students learn syntax, introduce common errors, model well-written code, and provide instant feedback [45]. These types of problems enable learners to demonstrate semantic and strategic knowledge without having to generate syntax, although some students use syntactic clues within the blocks to piece together the solution without necessarily understanding the problem, which can lead to a trial-and-error problem-solving strategy [14, 63]. These problems typically only have one correct solution, yet there are many ways to write code from scratch [45].

Parsons problems prompt the kind of explicit learning computer scientist have advocated for [55]. Explicit learning is facilitated by direct and unambiguous delivery of procedures and scaffolding to guide learners through the learning process with clear goals and ways to measure success [2]. Instructors have used Parsons problems as both formative and summative assessments [14]. Scores on Parsons problems correlate highly with scores on write-code assignments [7, 14]. Since Parsons and Haden's initial study, researchers have developed a variety of Parsons problems; they can vary by dimension, feedback, adaptation, and use of distractors. Parsons problems are also used to scaffold learning how to write code from scratch [26, 27]. Yet the extent to which Parsons problem research generalizes to programmers with disabilities is unknown.

2.2.1 Accessibility of Parsons Problems. Most research on teaching programming to learners with cognitive disabilities has focused on block-based programming—a popular approach used to increase inclusion and equity in the classroom because it's an appealing activity [12, 63]. Accessibility research on block-based programming has mostly focused on K-12 programmers with visual or motor impairments using the Scratch visual programming language [39, 40] and promising new research using Quorum Blocks [56]. Several challenges exist for learners with visual impairments, including navigating, comprehending, debugging, and skimming code [41]. However, block-based programming is distinct from Parsons problems in terms of the problem statement, the scope of the problem, and the design of the user interface [16]. Generally, environments for block-based programming are open-ended, in contrast to Parsons problems, which are not.

Parsons problems can feature two types of adaptation. Intra-problem adaptation happens when the difficulty of the current problem is dynamically reduced after three incorrect attempts. Each time the learner clicks the "Help Me" button to initiate the intra-problem adaptation, the system will remove a distractor block from the solution, or if no distractors are left and more than three blocks remain in the solution, it will combine two blocks into one. In addition, inter-problem adaptation happens when the difficulty of the next problem is changed based on the learner's performance on the previous problem. If the learner struggles, the next problem can be made easier by removing some or all of the distractor blocks. If they solved the previous problem in one attempt, the next problem could be made harder by using all the distractor blocks and showing them randomly mixed in with the correct blocks. These features can aid us in creating cognitively accessible learning experiences.

Parsons problem research has generally focused on what neurotypical individuals prefer (Parsons or write-code problems) and their computational practices, perspectives, and attitudes *inside* classrooms or labs (i.e., how they solve Parsons problems, whether they find them useful, and if they comprehend the adaptation process) [16]. Hence, this study explores the cognitive accessibility [see

31] of Parsons problems for neurodiverse programmers learning to code *outside* of the classroom.

3 METHODOLOGY

To answer the research question, I conducted an exploratory multiple-case study [62]. Case study methodology was originally intended for exploratory purposes [18]. Exploratory case studies are designed to discover what's happening, search for new insights, and generate ideas and working hypotheses for future research [62]. Case studies are also a popular research design of inquiry into how learners with cognitive disabilities learn how to program [12].

3.1 Participants

The author recruited novice programmers with disabilities from a postsecondary research institution in the northern Midwest of the United States via a flyer sent to several listservs. Participants were eligible if they had a cognitive, learning, or neurological disability as categorized by W3C's Web Accessibility Initiative (WAI) and defined by the Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-V). They were asked to fill out a sign-up questionnaire used to screen participants, which included a prior programming experience survey [25] and a disability questionnaire derived from [3] if they were eligible. Participants received a $500.00 stipend for completing the study. Five participants completed the study; demographic information about each participant is shown in Table 1; their ages ranged from 21 to 38.

3.2 Materials

The eBook used in this study is an interactive version of Dr. Charles Severance's *Python for Everybody*. It features typical instructional material (text, pictures, videos) and interactive features with immediate feedback (code-writing problems, debugging problems, Parsons problems, multiple-choice questions, fill-in-the-blank questions, and matching questions) [17]. The eBook covers programming fundamentals (strings, variables, loops, conditionals, functions), data structures (lists, tuples, dictionaries), object-oriented programming, and an introduction to data science (Files, Beautiful Soup, APIs, Databases). Participants were asked to complete eight sets of reading and programming practice problems (see Table 2). There were a total of 32 Parsons problems that had anywhere between zero and five distractor blocks. For an example of the problems used in this study see Figure 1.

3.3 Protocol

Participants were reminded via email once a week to complete the reading and practice assignments. The deadline for completing the study was flexible. Upon completing each week, participants were also asked to answer an open-ended question. It asked them to "Please read over the strategies, standards, and resources from W3C for making the web accessible to people with cognitive disabilities by clicking on this link and then answer the following question. Did you encounter any accessibility barriers and/or benefits while using the interactive eBook today? If so, please explain?"

I conducted three retrospective think-aloud interviews with each participant online via Zoom after they completed the reading and practice problems for weeks three, six, and eight. First, I asked participants for consent. Second, I asked them to solve a set of computer programming problems while I took notes to support my observations. Finally, I followed up with the participants about their responses to the open-ended question about the cognitive accessibility of Parsons problems.

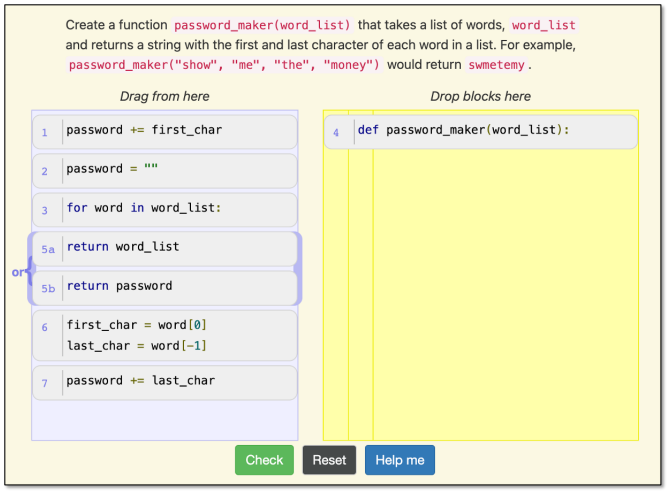

Figure 1: Example Parsons problems from Chapter Seven

3.4 Analysis

Multiple-case studies can be examined through both within-case analysis and cross-case synthesis [62]—the latter has been used in computing education research to understand the needs of as little as two participants [54]. Each of the retrospective think-aloud interviews were transcribed and then qualitative analysis was performed using ATLAS.ti. The author developed a codebook using a structural coding approach based on the research question: one code for accessibility barriers and one code for accessibility benefits [50]. The author and a colleague coded 20% of the transcripts and identified examples independently until we reached 100% agreement based on recommendations from [21]. The author coded the remaining transcripts independently.

4 RESULTS AND DISCUSSION

Each week, the participants were asked if they encountered any accessibility barriers or benefits while solving the Parsons problems at the end of each chapter and, if so, to explain. Within-subject analyses led to one design recommendation about increasing the cognitive accessibility of Parsons problems for each participant. Each participant picked their pseudonym.

4.1 Programmers with Seizure Disorders

Amanda reported experiencing focal seizures when solving Parsons problems due to the presence of numbers. Focal seizures can originate in the temporal, frontal, occipital, or parietal lobe [see 47]. In response to the question about the cognitive accessibility of Parsons problems, Amanda said:

Table 1: Participant Demographics

Pseudonym	Gender	Ethnicity	Disability	Programming Experience
Amanda (she/her)	Female	White	Mental Health Disability; Neurodiverse; Seizure Disorder	SPSS, R
Claire (she/her)	Female	White	Attention Deficit Hyperactivity Disorder (ADHD); Neurodiverse	SPSS, R
User (they/them)	Agender	Russian-Yakut	ADHD; Mental Health Disability	None
Sophia (she/her or they/them)	Female	Latina	Learning Disability; Mental Health Disability; Memory Impairment	Mplus, SPSS, R
John (he/him)	Male	Asian, White	Neurodiverse; Tourette Syndrome	None

Table 2: Weekly eBook Chapter Assignments with Estimated Reading Time

Week	Chapter	# of Pages	Estimated Reading Time
1	Variables, Expressions, and Statements	14	40 min.
2	Debugging	5	30 min.
3	Conditional Execution	11	40 min.
4	Functions	13	40 min.
5	Loops and Iterations	8	40 min.
6	Strings	13	40 min.
7	Lists	15	46 min.
8	Dictionaries	7	33 min.

Notes: The estimated reading time is based on 200 words per minute.

> *"I've had two seizures....The first three days, it didn't happen. I think [during the chapter on conditional execution and functions]. Numbers are sometimes triggering for me...usually, when I have a seizure, I'll put my head down to get blood back to my head. I normally feel kind of faint. I don't have convulsive seizures. I have focal seizures....It's not really anything that can be controlled. It's just the presence of numbers, and sometimes if I'm going through [the material] too fast. If it says it's going to take forty minutes, it'll probably take me two hours."*

This observation led to the following design consideration:

Learners with seizure disorders may experience more seizures when solving Parsons problems that require mental calculations.

While pace may be a factor, this finding is consistent with previous research on the relationship between mental arithmetic and seizures [61]. Ingvar and Nyman [29] termed this *epilepsia arithmetices* to describe a sort of reflex epilepsy in which mental arithmetic results in clinical seizures. Amanda also reported that her seizures come from her right temporal lobe, consistent with research on number processing in the temporal lobe and epilepsy [13].

4.2 Programmers with ADHD and Tourette Syndrome

Claire reported that distractor blocks improved her comprehension and helped with the inattention she experienced while solving Parsons problems.

During one of the think-aloud sessions, when solving a problem with three distractor blocks, Claire expressed that she was not preoccupied with them but focused and knew what to do (see Figure 1 for an example of distractor blocks paired using the word 'or'). She said that she was not distracted by them because she "didn't pay enough attention to them or how many there were, but rather, she paid attention to choosing the correct block. Claire commented,

> *"I actually think the distractors are beneficial because I have to think about the code itself rather than just the order...having the [distractor blocks] makes me think about which one of [the blocks] is the correct way to do this [the correct approach], so for the future, I've already thought about that and processed it a little more because, without it, I'm kind of prone to just glancing at it and saying, 'Oh, that looks right.' But then I didn't have to think about exactly what it is."*

Claire said she thought distractor blocks would be better for some learners with ADHD. She agreed with the statement that the distractor blocks caused her to focus and pay attention. Claire explained:

> *"because if I read something, sometimes I'll just read it for....this is an analogy, but I've learned a lot of foreign languages, and I've found when I'm trying to learn the vocabulary and I just look at a word, I'll understand what the word means. But then if I have to recall it, I won't know how to spell it correctly....the [distractors] make me think about the spelling or how it's written. And then I understand it better. I actually have to look at it and think about it."*

Similarly, John, who openly identified as having Tourette Syndrome and being neurodiverse, expressed a preference for paired distractor blocks —blocks linked by 'OR'—due to his eye and motor tics. John said, "...with my Tourette syndrome, it takes extra time to type, and moving things around using a mouse takes a lot longer for me just because there are interruptions."

This observation led to the following design consideration:

> Learners with an attention-deficit/hyperactivity disorder (ADHD) or Tourette syndrome may learn more from Parsons problems with *paired* versus jumbled distractor blocks.

Attention-deficit/hyperactivity disorder (ADHD) is frequently diagnosed in people with Tourette syndrome [57]. Learners such as Claire, who have ADHD, experience an increase in distractibility [1]. "Inattention involves difficulties with keeping the learner's attention focused and to shift the focus of attention as necessary" [46, p. 75]. Similarly, programmers with Tourette syndrome, like John, experience impaired motor control, which John identified as an interruption/distraction when learning. Distractor blocks are extra code blocks not part of a correct Parsons problem solution. Whereas past researchers have found distractors increased extraneous cognitive load [22] and time-on-task [53], the present study provides evidence that *paired* distractor blocks may improve focus for learners with ADHD or Tourette syndrome. This would be a desirable difficulty [cf. 6]. Unpaired distractors may not focus the attention of learners with ADHD and not support efficient drag-and-drop actions for learners with Tourette syndrome who experience tics—"sudden, habit-like movements or utterances that typically mimic some fragment of normal behavior and involved discrete muscle groups" [57, p.956]. Moreover, researchers who have investigated how learners with ADHD learn introductory computer programming concepts state one approach is to use completed examples [46] of which Parsons problems can be considered a variant [16].

4.3 Programmers with a Mental Health Disability

User had stated that they had a mental health disability, so I followed up with them about it. When asked what kinds of positive content or activities would raise their energy and mood, User responded:

> *"One of the things that I do for emotional regulation is look at videos of cats....I would [also] say empowering social justice content like an article about a social work problem or social media content in which a person understands how taking care of themselves is actually resistance and a revolutionary practice. There's a whole range of things that feel affirming....a social justice program—people would be all over that—and gender fluidity. I don't think that kind of thing exists though..."*

User reported that they watched videos of cats to regulate their emotions while learning and that they would be motivated by social justice-oriented, relevant, interest-driven Parsons problems. This observation led to the following design consideration:

> Learners with an attention-deficit/hyperactivity disorder (ADHD) or a mental health disability may experience an increase in (1) focus or (2) positive emotions if presented with relevant or interest-driven Parsons problems.

Computing education researchers have found there is a need to support self-regulated learning (SRL) strategies such as emotional regulation in computer science education (CSEd) [20] and to design "intelligent systems that respond adaptively to students' emotions" when learning how to program [11, p. 30]. Kinnunen et al. [32] used a media computation approach to explore learners' emotional experiences with Java programming assignments and found that students with negative programming experiences reflect positively on their self-efficacy. Learning scientists have also posited that "because learning is influenced in fundamental ways by the context in which it takes place, schools and classrooms should be learner and community-centered" [42, p. 22]. Context and culture are critical aspects of learning. And furthermore, research shows that participants like User and Sophia, who struggled with math problems, benefit from intelligent tutoring systems that personalize problems to students' out-of-school interests [59].

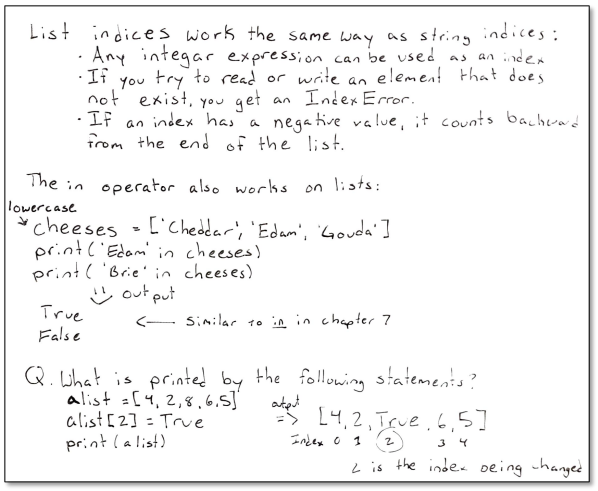

Figure 2: Sophia's Handwritten Notes on Lists

4.4 Programmers with Memory Impairment

Sophia's memory was impaired due to a traumatic brain injury. She reported that longer and more complex chapters in the eBook required note-taking and that she interacted better with text on paper than eBooks because of her memory impairment. She and Claire took copious notes and used them to solve Parsons problems. In particular, when solving problem two during the last think-aloud session, Sophia used her notes on lists to help (see Figure 2). This observation led to the following design consideration:

> Learners with memory impairment or ADHD may need easy access to their notes to increase their problem-solving performance.

This observation is consistent with research linking note-taking, memory, and comprehension and the usefulness of note-taking and concept mapping in introductory computer science courses [30]. Note-taking is an active learning strategy that improves problem-solving performance [58]. Sophia engaged in both *constructive* (free-form) and *active* (summarized/copy-and-pasted) note-taking [8].

4.4.1 Limitations. First, this study only examined five novice programmers with different and some overlapping disabilities; one cannot assume these findings will generalize to other contexts. A replication of this study with a larger sample of learners who identify with each of the participants' disabilities would further support each corresponding design recommendation and whether these findings can be generalized across learners and contexts.

Second, participants in this study were not exposed to other environments for solving Parsons problems and learning how to program; hence, the results may be a byproduct of the interactive eBook used in this study.

4.4.2 Future Work. Future investigations into the effectiveness of Parsons problems for learners with disabilities should compare their experiences with different environments that support Parsons problems. Various programming practice websites and open-source tools support instructors and learners in using Parsons problems including Codio, Codespec, Epplets, js-parsons, PraireLearn, Runestone Academy, and UPP (the Unnamed Parsons Problem Tool). Furthermore, human-computer interaction (HCI) research is rich with measures such as Fitts' law that may help us better investigate human movement and accessibility concerns [37].

Disciplines offer us an excellent way to be sensitive to variations in learning [4], and computing education instructors and researchers have called for us to increase computing in other disciplines (also known as CS + X) [51]. Future research should investigate how we can develop discipline-specific programming problems in Python and other languages that increase cognitive relevance [10] and the quality of the overall learning experience.

Finally, future research on Parsons problems should explore paradigms such as Grid-Coding, speech-driven programming, and the use of problem-solving stages to improve accessibility for neurodiverse programmers, sighted, blind, and low-vision (BLV) programmers, and programmers with motor impairments [15, 43].

5 CONCLUSION

With the appropriate scaffolding and support, neurodiverse learners can excel at computer programming. This work contributed to the first empirical multi-case study on how neurodiverse learners learn how to program *outside* of the classroom with an interactive eBook with Parsons problems. Think-aloud observations led to the generation of five design considerations about how to improve the cognitive accessibility of Parsons problems. First, there exists a relationship between mental arithmetic and clinical seizures that should inform how instructors and cognitive tutors decide which computer programming problems to present to novice programmers. Second, this work highlights the impact that paired vs. jumbled distractor blocks may have on the learning experience for programmers with disabilities. This has implications for how we create these kinds of blocks and when we should and should not present them to learners (i.e., we would not want to make it harder for learners with tics to choose between these blocks). This work also adds to the findings on emotional regulation in computer science education concerning its importance in the design of computer programming problems. And, providing a space for note-taking may improve the problem-solving performance of programmers with disabilities. Future research should explore the generalizability of such observations with larger groups of learners who identify with each case.

ACKNOWLEDGMENTS

I thank Aadarsh Padiyath for their help in analyzing the think-aloud observations. This research was funded by the Rackham Graduate School and Rackham Fellowships Office at the University of Michigan.

REFERENCES

[1] Rebecca Adams, Paul Finn, Elisabeth Moes, Kathleen Flannery, and Albert Skip Rizzo. 2009. Distractibility in Attention/Deficit/ Hyperactivity Disorder (ADHD): the virtual reality classroom. *Child Neuropsychol.* 15, 2 (March 2009), 120–135.

[2] Anita L Archer and Charles A Hughes. 2010. *Explicit Instruction: Effective and Efficient Teaching*. Guilford Publications.

[3] Brianna Blaser and Richard E Ladner. 2020. Why is Data on Disability so Hard to Collect and Understand?. In *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, Vol. 1. ieeexplore.ieee.org, 1–8.

[4] R Brooks. 2011. Making the case for discipline-based assessment. *Literary Study, Measurement, and the Sublime* (2011).

[5] Robert Chapman. 2020. Defining neurodiversity for research and practice. In *Neurodiversity studies*. Routledge, 218–220.

[6] Ouhao Chen, Juan C Castro-Alonso, Fred Paas, and John Sweller. 2018. Undesirable Difficulty Effects in the Learning of High-Element Interactivity Materials. *Front. Psychol.* 9 (Aug. 2018), 1483.

[7] Nick Cheng and Brian Harrington. 2017. The Code Mangler: Evaluating Coding Ability Without Writing any Code. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 123–128.

[8] Michelene T H Chi and Ruth Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educ. Psychol.* 49, 4 (Oct. 2014), 219–243.

[9] Lynn Clouder, Mehmet Karakus, Alessia Cinotti, María Virginia Ferreyra, Genoveva Amador Fierros, and Patricia Rojo. 2020. Neurodiversity in higher education: A narrative synthesis. *Higher Education* 80, 4 (2020), 757–778.

[10] Erica Cosijn and Peter Ingwersen. 2000. Dimensions of relevance. *Inf. Process. Manag.* 36, 4 (July 2000), 533–550.

[11] Mayela Coto, Sonia Mora, Beatriz Grass, and Juan Murillo-Morera. 2022. Emotions and programming learning: systematic mapping. *Computer Science Education* 32, 1 (Jan. 2022), 30–65.

[12] Elaine Cristina Juvino De Araújo and Wilkerson L Andrade. 2021. A Systematic Literature Review on Teaching Programming to People with Cognitive Disabilities. In *2021 IEEE Frontiers in Education Conference (FIE)*. ieeexplore.ieee.org, 1–8.

[13] M Delazer, A Gasperi, L Bartha, E Trinka, and T Benke. 2004. Number processing in temporal lobe epilepsy. *J. Neurol. Neurosurg. Psychiatry* 75, 6 (June 2004), 901–903.

[14] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the Fourth international Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08)*. Association for Computing Machinery, New York, NY, USA, 113–124.

[15] Md Ehtesham-Ul-Haque, Syed Mostofa Monsur, and Syed Masum Billah. 2022. Grid-Coding: An Accessible, Efficient, and Structured Coding Paradigm for Blind and Low-Vision Programmers. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22, Article 44)*. Association for Computing Machinery, New York, NY, USA, 1–21.

[16] Barbara J Ericson, Paul Denny, James Prather, Rodrigo Duran, Arto Hellas, Juho Leinonen, Craig S Miller, Briana B Morrison, Janice L Pearce, and Susan H Rodger. 2022. Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (Dublin, Ireland) *(ITiCSE-WGR '22)*. Association for Computing Machinery, New York, NY, USA, 191–234.

[17] Barbara J Ericson and Bradley N Miller. 2020. Runestone: A Platform for Free, Online, and Interactive Ebooks. In *Proceedings of the 51st ACM Technical Symposium*

on Computer Science Education (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1012–1018.
[18] Bent Flyvbjerg. 2006. Five Misunderstandings About Case-Study Research. *Qual. Inq.* 12, 2 (April 2006), 219–245.
[19] Flynn Fromont, Hiruna Jayamanne, and Paul Denny. 2023. Exploring the Difficulty of Faded Parsons Problems for Programming Education. In *Proceedings of the 25th Australasian Computing Education Conference* (Melbourne, VIC, Australia) *(ACE '23)*. Association for Computing Machinery, New York, NY, USA, 113–122.
[20] Rita Garcia, Katrina Falkner, and Rebecca Vivian. 2018. Systematic literature review: Self-Regulated Learning strategies using e-learning tools for Computer Science. *Comput. Educ.* 123 (Aug. 2018), 150–163.
[21] David Hammer and Leema K Berland. 2014. Confusing Claims for Data: A Critique of Common Practices for Presenting Qualitative Research on Learning. *Journal of the Learning Sciences* 23, 1 (Jan. 2014), 37–46.
[22] Kyle James Harms, Jason Chen, and Caitlin L Kelleher. 2016. Distractors in Parsons Problems Decrease Learning Efficiency for Young Novice Programmers. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 241–250.
[23] Carl C Haynes and Barbara J Ericson. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21, Article 60)*. Association for Computing Machinery, New York, NY, USA, 1–15.
[24] Carl Haynes-Magyar and Barbara Ericson. 2022. The Impact of Solving Adaptive Parsons Problems with Common and Uncommon Solutions. In *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '22, Article 23)*. Association for Computing Machinery, New York, NY, USA, 1–14.
[25] Edward Holden and Elissa Weeden. 2003. The impact of prior experience in an information technology programming course sequence. In *Proceedings of the 4th conference on Information technology curriculum* (Lafayette, Indiana, USA) *(CITC4 '03)*. Association for Computing Machinery, New York, NY, USA, 41–46.
[26] Xinying Hou, Barbara Jane Ericson, and Xu Wang. 2022. Using Adaptive Parsons Problems to Scaffold Write-Code Problems. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22, Vol. 1)*. Association for Computing Machinery, New York, NY, USA, 15–26.
[27] Xinying Hou, Barbara J Ericson, and Xu Wang. 2023. Understanding the Effects of Using Parsons Problems to Scaffold Code Writing for Students with Varying CS Self-Efficacy Levels. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '23, Vol. 1)*. Association for Computing Machinery, New York, NY, USA, 1–12.
[28] Office of Special Education Programs Individuals with Disabilities Education Act (IDEA). 2021. U.S. Department of Education.
[29] D H Ingvar and G E Nyman. 1962. Epilepsia arithmetices: A new psychologic trigger mechanism in a case of epilepsy. *Neurology* 12, 4 (April 1962), 282–282.
[30] Taina Kaivola and Heikki Lokki. 2010. Using concept mapping as a note taking tool in computer science. https://helda.helsinki.fi/bitstream/handle/10138/27360/cmc2010_a27.pdf?sequence=2. Accessed: 2023-5-3.
[31] Terhi Kärpänen. 2021. A Literature Review on Cognitive Accessibility. *Stud. Health Technol. Inform.* 282 (June 2021), 259–270.
[32] Päivi Kinnunen and Beth Simon. 2012. My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Comput. Sci. Educ.* 22, 1 (March 2012), 1–28.
[33] Varsha Koushik and Shaun K Kane. 2019. "It Broadens My Mind": Empowering People with Cognitive Disabilities through Computing Education. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19, Paper 514)*. Association for Computing Machinery, New York, NY, USA, 1–12.
[34] Richard E Ladner and Maya Israel. 2016. "For all" in "computer science for all". *Commun. ACM* 59, 9 (Aug. 2016), 26–28.
[35] Christopher Luchs. 2021. Considering Neurodiversity in Learning Design and Technology. *TechTrends* 65, 6 (Nov. 2021), 923–924.
[36] Kelly Mack, Emma McDonnell, Dhruv Jain, Lucy Lu Wang, Jon E. Froehlich, and Leah Findlater. 2021. What Do We Mean by "Accessibility Research"? A Literature Survey of Accessibility Papers in CHI and ASSETS from 1994 to 2019. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21, Article 371)*. Association for Computing Machinery, New York, NY, USA, 1–18.
[37] I Scott MacKenzie. 2017. Fitts' Law. In *The Wiley Handbook of Human Computer Interaction*. John Wiley & Sons, Ltd, Chichester, UK, 347–370.
[38] Sharan B Merriam. 1995. N of I?: Issues of Validity and Reliability in. *PAACE Journal of lifelong learning* 4 (1995), 51–60.
[39] Lauren R Milne and Richard E Ladner. 2019. Position: Accessible Block-Based Programming: Why and How. In *2019 IEEE Blocks and Beyond Workshop (B&B)*. ieeexplore.ieee.org, 19–22.
[40] Aboubakar Mountapmbeme and Stephanie Ludi. 2021. How Teachers of the Visually Impaired Compensate with the Absence of Accessible Block-Based Languages. In *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility* (Virtual Event, USA) *(ASSETS '21, Article 4)*. Association for Computing Machinery, New York, NY, USA, 1–10.
[41] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. 2022. Addressing Accessibility Barriers in Programming for People with Visual Impairments: A Literature Review. *ACM Trans. Access. Comput.* 15, 1 (March 2022), 1–26.
[42] National Academies of Sciences, Engineering, and Medicine, Division of Behavioral and Social Sciences and Education, Board on Science Education, Board on Behavioral, Cognitive, and Sensory Sciences, and Committee on How People Learn II: The Science and Practice of Learning. 2018. *How People Learn II: Learners, Contexts, and Cultures*. National Academies Press.
[43] Obianuju Okafor and Stephanie Ludi. 2022. Voice-Enabled Blockly: Usability Impressions of a Speech-driven Block-based Programming System. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility* (Athens, Greece) *(ASSETS '22, Article 64)*. Association for Computing Machinery, New York, NY, USA, 1–5.
[44] Stack Overflow. 2022. Stack Overflow Developer Survey 2022.
[45] Parsons and Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. *of the 8th Australasian Conference on . . .* (2006).
[46] Colin Pilkington and Helene Gelderblom. 2010. Using the Karplus learning cycle to teach learners with ADHD introductory computer programming. *African Journal of Research in Mathematics, Science and Technology Education* 14, 1 (Jan. 2010), 73–84.
[47] Colin Reilly and Rebecca Ballantine. 2011. Epilepsy in school-aged children: more than just seizures? *Support Learn.* 26, 4 (Nov. 2011), 144–151.
[48] Scott Michael Robertson. 2010. Neurodiversity, Quality of Life, and Autistic Adults: Shifting Research and Professional Focuses onto Real-Life Challenges. *Disabil. Stud. Q.* 30, 1 (2010).
[49] Hanna Bertilsdotter Rosqvist, Nick Chown, and Anna Stenning. 2020. *Neurodiversity Studies: A New Critical Paradigm*. Routledge.
[50] Johnny Saldaña. 2021. The coding manual for qualitative researchers. *The coding manual for qualitative researchers* (2021), 1–440.
[51] Devin W Silvia, Marcos D Caballero, Thomas Finzell, Rachel Frisbie, Patti Hamerski, Emily Bolger, Sarah Castle, Rachel Roca, and Paige Tourangeau. 2023. Computing in Support of Disciplinary Learning. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1247.
[52] Judy Singer. 2017. *Neurodiversity: The birth of an idea*.
[53] David H Smith, IV and Craig Zilles. 2023. Discovering, autogenerating, and evaluating distractors for python parsons problems in CS1. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15â•fi18, 2023, Toronto, ON, Canada* 1, 1 (2023).
[54] Melinda R Snodgrass, Maya Israel, and George C Reese. 2016. Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Comput. Educ.* 100 (Sept. 2016), 1–17.
[55] Elliot Soloway. 1986. Learning to program= learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.
[56] Andreas Stefik, Willliam Allee, Gabriel Contreras, Timothy Kluthe, Alex Hoffman, Brianna Blaser, and Richard Ladner. [n. d.]. Accessible to Whom? Bringing Accessibility to Blocks. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20â•fi23, 2024, Portland, OR, USA* 1, 1 ([n. d.]).
[57] James E Swain, Lawrence Scahill, Paul J Lombroso, Robert A King, and James F Leckman. 2007. Tourette syndrome and tic disorders: a decade of progress. *J. Am. Acad. Child Adolesc. Psychiatry* 46, 8 (Aug. 2007), 947–968.
[58] J Gregory Trafton and Susan B Trickett. 2001. Note-Taking for Self-Explanation and Problem Solving. *Human–Computer Interaction* 16, 1 (March 2001), 1–38.
[59] Candace Walkington and Matthew L Bernacki. 2019. Personalizing Algebra to Students' Individual Interests in an Intelligent Tutoring System: Moderators of Impact. *International Journal of Artificial Intelligence in Education* 29, 1 (March 2019), 58–88.
[60] Nathaniel Weinman, Armando Fox, and Marti A Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21, Article 53)*. Association for Computing Machinery, New York, NY, USA, 1–4.
[61] A J Wilkins, B Zifkin, F Andermann, and E McGovern. 1982. Seizures induced by thinking. *Ann. Neurol.* 11, 6 (June 1982), 608–612.
[62] Robert K Yin. 2018. *Case study research and applications*. Sage.
[63] Rui Zhi, Min Chi, Tiffany Barnes, and Thomas W Price. 2019. Evaluating the Effectiveness of Parsons Problems for Block-based Programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 51–59.