Exploring Motor-impaired Programmers' Use of Speech Recognition

Sadia Nowrin
snowrin@mtu.edu
Michigan Technological University
Houghton, Michigan, USA

Patricia Ordóñez
patricia.ordonez@upr.edu
University of Puerto Rico Río Piedras
San Juan, Puerto Rico

Keith Vertanen
vertanen@mtu.edu
Michigan Technological University
Houghton, Michigan, USA

ABSTRACT

Typing programs can be difficult or impossible for programmers with motor impairments. Programming by voice can be a promising alternative. In this research, we explored the perceptions of motor-impaired programmers with regard to programming by voice. We learned that leveraging existing voice-based programming platforms to speak code can be more complicated than it needs to be. The interviewees expressed their frustration with long hours of memorizing unnatural commands in order to enter code by voice. In addition, we found a preference for being able to speak code in a flexible manner without requiring strict adherence to a grammar.

CCS CONCEPTS

• **Human-centered computing → Accessibility**.

KEYWORDS

Voice Programming, Speech Recognition, Voice User Interfaces, Accessibility

ACM Reference Format:
Sadia Nowrin, Patricia Ordóñez, and Keith Vertanen. 2022. Exploring Motor-impaired Programmers' Use of Speech Recognition. In *The 24th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '22), October 23–26, 2022, Athens, Greece.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3517428.3550392

1 INTRODUCTION

Professionals from diverse disciplines in both industry and academia are forced to constantly type [6]. Programmers mostly rely on a keyboard and a mouse to input programs. Such reliance on a keyboard and a mouse may create a barrier for people with disabilities to learning or successfully pursuing careers in computer science or related fields. In a 2022 survey of 70,000 Stack Overflow developers, 0.35% said they could not or had difficulty typing [7]. Practicing software engineers with no motor impairments may also develop acquired conditions such as Repetitive Strain Injury (RSI) as a result of constantly typing. An individual with RSI may lose the ability to type entirely but their cognitive ability and desire to program may remain unaffected.

Software developers with motor impairments have created systems for their own use and for those willing to devote a large amount of effort to learn a new spoken language. Tavis Rudd, a software developer with RSI built a system that allowed him to write code by voice. Tavis Rudd's spoken Python language [10] requires users to learn a complex set of over 1000 commands. After developing severe hand pain, Ryan Hileman left his full-time job as a software engineer and started developing Talon[1]. Talon is a freeware software that enables hands-free input via speech recognition and eye tracking. Both Talon and Tavis Rudd's systems do not have the ability to perform natural language processing and the commands have to match exactly with the system's commands. After being diagnosed with RSI, Rick Mohr built Vocola[2], a spoken command language to control a computer. Serenade[3], a leading voice programming platform nowadays, was created out of need when co-founder Matt Wiethoff was diagnosed with RSI. Serenade also requires that spoken commands precisely match the commands that the system understands. However, their techniques can be significantly more productive than typing for users willing to invest time learning the commands.

Prior research has explored how programmers might code by voice when recognition is performed by a human (i.e. a Wizard of Oz experiment) [8, 9], or by asking programmers to read pre-written code aloud [1]. Researchers have also developed voice-based systems to write code that required learning a complex command language [1, 3–5, 9, 11]. According to an evaluation of a voice programming system [2], programmers needed extensive training to master the commands. We think, rather than attempt to build a working interface, the very first step towards developing a robust voice programming system is to understand target users' needs and preferences. Our paper serves to investigate the preferences of seven programmers who depend on voice programming interfaces for their careers. We explored their thoughts on current voice programming systems, the difficulties they encountered while using voice to enter programs, and what they expect from a future voice programming system.

2 INTERVIEWS WITH MOTOR-IMPAIRED PROGRAMMERS

We conducted semi-structured interviews with seven programmers with motor impairments. Our goal was to understand the perceptions of programmers with disabilities. All participants were native English speakers. Table 1 presents the details about the programmers with motor impairments we interviewed. Programming

ASSETS '22, October 23–26, 2022, Athens, Greece
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9258-7/22/10.
https://doi.org/10.1145/3517428.3550392

[1]https://talonvoice.com/
[2]http://vocola.net/
[3]https://serenade.ai/

Participant	Gender	Current job	Physical condition	Programming Experience
P1	female	graduate student	chronic musculoskeletal pain	8 years
P2	male	web developer	neurological accident	6-7 years
P3	female	graduate student	spinal muscular atrophy	20 years
P4	female	academic researcher	small fiber neuropathy	16 years
P5	male	professor	congenital upper limb deficiency	17 years
P6	male	academic researcher	upper limb musculoskeletal disorder	6 years
P7	male	software architect	temporary repetitive strain injury	24 years

Table 1: Details about the programmers with motor impairments we interviewed.

experience ranged from 6–20 years. All participants wrote programs almost everyday. All used a combination of mouse, keyboard, and voice to program. All used voice user interfaces for tasks like writing emails and getting directions.

2.1 Study Design

The study consisted of a questionnaire followed by a set of open-ended questions. The questionnaire included demographic questions, participants' experience with voice user interfaces, and their programming experience. See our supplementary materials for the exact interview questions. We asked participants to rate their experience on a 7-point Likert scale. For the open-ended questions, we first asked participants to describe their experience with voice user interfaces for tasks other than programming. Then they spoke about their experience of using voice programming tools as well as the difficulties they encountered in entering programs by voice. Additionally, we asked participants how they imagine writing a program using an intelligent and highly accurate future voice programming system. Participants also explained what they thought would be the most challenging part of programming by voice. Finally, participants were asked whether they have any privacy or social concerns regarding using a voice programming system, and in what circumstances they thought programming by voice would be useful. We conducted the interviews via videoconferencing software. Participants were paid $20 for taking part in the study.

2.2 Data Analysis

We transcribed the interview recordings and read the transcripts repeatedly. To better understand the participants' perspectives, we employed thematic analysis on the interview data. We first derived codes from the interview data and then grouped the similar codes together into broader themes. The resulting themes helped us identify significant information in the data. Finally, we revisited the transcripts to make sure whether the themes accurately reflected the entire data and the study's objective.

3 RESULTS

From the quantitative analysis of the questionnaire data, we found six participants used voice interfaces for tasks such as writing emails and getting directions. Two participants strongly agreed and four participants agreed that speech interfaces sometimes had trouble understanding them. Five participants strongly agreed that they were expert programmers. Four participants agreed and three participants strongly agreed that they frequently wrote programs.

From the thematic analysis of the qualitative data, we found five themes: experience, vision, challenges, privacy, and usefulness. The resulting themes along with the description and the corresponding quotes are presented in Table 2.

Theme: Experience. All participants indicated they took frequent breaks while writing programs. All participants used both Dragon Naturally Speaking and Voice Code. Three of them currently use Talon Voice to dictate programs. One participant said that existing voice programming technologies can dictate simple words accurately like print, insert tabs, and add punctuation like single and double quotes. Two participants stated that the technologies they used to program by voice were not well documented. In addition, the interviewees mentioned that the existing systems forced them to learn a large number of commands. Learning these commands required many hours and the process was frustrating and stressful. Two participants said that the system occasionally misrecognized a very long command and they found it annoying and time-consuming to input the command again.

Theme: Vision. Five participants mentioned that they wish they could write programs via natural language. They also mentioned that there should be a mechanism to distinguish between words that sound the same in English but signify different things in the context in which they were stated (e.g. the word "to" and the number "two"). Participants thought an intelligent voice programming system should be able to incorporate corrections. According to one participant, it would be useful to be able to navigate through the code and edit a specific word on a certain line.

Theme: Challenges. Six participants thought variable names would be very challenging to dictate as they are sometimes not normal spoken words. Two participants expressed that dictating new variables is hard and they wish there were a mechanism to save all previously used variable names for faster dictation. Two participants talked about writing comments: dictating a bunch of symbols in the code and then switching to English to dictate comments was a real challenge. Indentation would be a big challenge while writing Python code according to three participants. Participants pointed out that dictating code from scratch and editing code by voice that already exists are two distinct things. The majority of the participants mentioned that navigating through the code and correcting errors would be challenging.

Theme: Privacy. Two participants were concerned about using a cloud-based recognizer as it's uncertain how the data would be used and who would access it. Two other participants mentioned they would be concerned about privacy if they had to speak

Themes	Description	Illustrative quotes
Experience	Experience of using voice user interfaces to write code	"It's certainly not doing natural language processing in interpreting your commands but they try to make them feel like a natural command, and I think sometimes they do it at the expense of the power and unambiguous nature that you could get." (P1) "There's no documentation, for what does what." (P2) "A lot of people bounce off of dictating code when they start because it's really annoying and unpleasant to learn all these commands." (P4)
Vision	Vision of a future voice programming system	"I'm going to assume something closer to like the Star Trek computer that has a lot of natural language processing." (P2) "I guess it would be more similar to the experience of pair programming with someone." (P4) "I wish we had a dynamically typed language that is processing in the background and telling you where the errors are appearing as you're writing the code and you can jump to the errors relatively easily." (P5)
Challenges	Parts of program difficult to input by voice	"Switching between comments and the rest of your code is I think a little hard because those are really two different modes." (P4) "There are some real challenges with variable names that might not be normal spoken word." (P1)
Privacy	Privacy or social concerns of using a voice programming interface	"I have privacy concerns about using voice interfaces when I'm not in a private place and can be overheard." (P1) "I really wouldn't want a voice interface that's like living in the cloud." (P4) "I would never use a cloud-based speech recognizer to write programs as it's hard to know what information is gathered and who might have access to it." (P6)
Usefulness	Circumstances in which programming by voice would be most useful	"People won't be intimidated by the task of typing by hand, so it would give them more energy and focus on actually solving the problem" (P3) "For people who are on the verge of developing RSI and also for people who can't type in the first place." (P7) "I would probably continue to use a voice interface, even if I'm recovered and could use a keyboard all the time and had no pain as I can just like lean back in my chair during easy things that I'm talking through." (P4)

Table 2: Thematic analysis on the interview data.

programs in a place where others could hear them. Others said they wouldn't be concerned about privacy as long as the data is anonymous.

Theme: Usefulness. All participants thought programming by voice would be useful for people with motor impairments like themselves. Two participants stated that programming by voice would also be useful for people with no motor impairments as they can relax while speaking easy parts of a program. Another participant mentioned that programming by voice would be useful for people who program as part of their job as they could focus more on problem-solving. According to the participants, programmers who do not have RSI but slowly developing it need to type a bit less and programming by voice could be very useful for them.

4 DISCUSSION

Most of the programmers with motor impairments we interviewed found today's speech recognition technology helpful in certain contexts such as writing emails. But this did not extend to more specialized contexts such as creating content in markup languages such as HTML and LaTeX or creating programs in languages such as Python. In cases where interviewees were using speech for such tasks, the solutions employed were awkward and required substantial training; they adapted to the limitations of the technology rather than the other way around. There is plenty of data available to train a speech recognizer that understands natural language, but very limited data exists to train a recognizer that understands programs. Further research is required to investigate how to collect a large number of spoken programs to train a speech recognizer that would be good at recognizing code.

Learning complex commands might make it difficult to learn how to program by voice. It may also create a barrier for novice programmers trying to learn to program, e.g. in an introductory programming course. One effective way to get around this problem is to input programs using natural language. We believe improving the language model used by the speech recognizer would be an essential step to supporting naturally spoken code input.

Programming is not only about writing code. Programmers must be able to navigate through the code, edit the code, debug their code and correct the errors. Making these interactions accessible to people with diverse motor abilities will be challenging. While current voice-based systems focus on inputting code from scratch, further study is needed to understand voice based debugging and code correction.

In theory, programming languages are more predictable than natural languages because they have strict grammar rules. But this gets complicated by user-defined names such as variable names and function names. Variable names may contain abbreviated words and mixed capitalization which makes them less predictable. Furthermore, it is not apparent whether programmers will speak a line of code using its exact syntax, or more naturally. A key step in developing an intelligent voice programming system would be to investigate how programmers speak various programs and user-defined names.

A 2018 article in Nature described two motor-impaired programmers' experience of using existing voice programming systems [6]. Our interview study of seven motor-impaired programmers provides new insights into some of the significant challenges in programming by voice (e.g. speaking variable names and editing code), as well as the type of future system that motor-impaired programmers envision.

5 CONCLUSIONS

In conclusion, our interviews revealed that, despite the existence of several voice programming systems, motor-impaired programmers had a desire for a more natural and purpose-built solution. It also highlighted their frustrations with the accuracy of current systems, as well as the challenges in dictating code. While the challenges are many, we think a practical and naturally spoken programming system would be valuable to many and should be feasible given recent advances in speech and natural language processing.

REFERENCES

[1] Andrew Begel and Susan L Graham. 2005. Spoken programs. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. IEEE, 99–106.

[2] Andrew Begel and Susan L Graham. 2006. An Assessment of a Speech-Based Programming Environment. In *Visual LAnguages and Human-Centric Computing (VL/HCC'06)*. IEEE, 116–120.

[3] Alain Désilets. 2001. VoiceGrip: A Tool for Programming-by-Voice. *International Journal of Speech Technology* 4, 2 (2001), 103–116.

[4] Rinor S Maloku and Besart Xh Pllana. 2016. HyperCode: Voice aided programming. *IFAC-PapersOnLine* 49, 29 (2016), 263–268.

[5] Jonathan Giovanni Soto Muñoz, Arturo Iván de Casso Verdugo, Eliseo Geraldo González, Jesús Andrés Sandoval Bringas, and Miguel Parra Garcia. 2019. Programming by Voice Assistance Tool for Physical Impairment Patients Classified in to Peripheral Neuropathy Centered on Arms or Hands Movement Difficulty. In *2019 International Conference on Inclusive Technologies and Education (CONTIE)*. IEEE, 210–2107.

[6] Anna Nowogrodzki. 2018. Speaking in code: how to program by voice. *Nature* 559, 2 (2018), 141–142.

[7] Stack Overflow. 2022. Stack Overflow Developer Survey 2022. https://survey.stackoverflow.co/2022.

[8] David E Price, DA Dahlstrom, Ben Newton, and Joseph L Zachary. 2002. Off to See the Wizard: using a" Wizard of Oz" study to learn how to design a spoken language interface for programming. In *32nd Annual Frontiers in Education*, Vol. 1. IEEE, T2G–T2G.

[9] Lucas Rosenblatt, Patrick Carrington, Kotaro Hara, and Jeffrey P Bigham. 2018. Vocal Programming for People with Upper-Body Motor Impairments. In *Proceedings of the Internet of Accessible Things*. ACM, 30.

[10] Tavis Rudd. 2013. Using Python to Code by Voice. http://pyvideo.org/video/1735/using-python-to-code-by-voice.

[11] Jessica Van Brummelen, Kevin Weng, Phoebe Lin, and Catherine Yeo. 2020. CONVO: What does conversational programming need?. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.