Voice Coding Experiences for Developers with Physical Impairments

Chris Creed and Sayan Sarcar
HCI Research Centre, Birmingham City University (UK), B4 7XG

Corresponding Author: chris.creed@bcu.ac.uk

Published Version: https://doi.org/10.1108/JET-02-2024-0021

Research funding – This work was supported through a grant from Google's Award for Inclusion Research Program.

Abstract

Purpose
This research explores the current working practices of voice coders with physical impairments and investigates their perceptions of a research prototype utilising a fixed grammar speech coding approach.

Design/methodology/approach
Semi-structured interviews were initially conducted with seven voice coders with physical impairments to understand their development practices, followed by an exploratory user study with five disabled voice coders to obtain their feedback on a fixed grammar voice coding system.

Findings
Interviews provided new insights around the tools voice coders utilise, the need for multimodal coding approaches, as well as experiences in working within mixed ability development teams. Findings from the user evaluation elicited views around the need for distinct monosyllable voice commands, as well as other requirements to support efficient voice coding (e.g., command chaining, intuitive navigation, and custom command definition).

Research limitations/implications
The research was conducted with experienced voice coders which may have influenced perceptions of the voice coding prototype. The system was also evaluated over a single evaluation session, whereas longitudinal research with novice voice coders could also present additional insights.

Practical implications
Further research is required into customisable multimodal voice coding approaches for developers with physical impairments who may have varying levels of experience in coding via alternative methods.

Originality/value
The findings present new insights around the working practices and unique requirements of voice coders with physical impairments and highlights important new avenues for future research.

Keywords – Voice coding, Accessibility, Assistive technology, Inclusive Design, Voice Interaction, Disabled Developers

Article classification – Research Paper

Introduction

People with physical impairments who experience challenges in utilising traditional mouse and keyboard devices can be excluded from embarking on careers within software engineering domains. One promising solution is the use of alternative input approaches such as voice coding which enables hands-free control of computers to write code using speech input (Begel & Graham, 2006; Rosenblatt et al., 2018). A number of studies in the literature have been exploring different voice coding approaches with a particular emphasis on the use of fixed grammars that utilise natural commands such as “type [code]”, “comment [text]”, and “select [syntax]” (Paudyal et al., 2020), although these approaches have not yet been fully integrated within mainstream development environments. Furthermore, these types of approaches have not been widely evaluated by disabled developers with experience of voice coding. It is therefore unclear how professional developers perceive these approaches and whether they present any benefits over current development methods. To address this gap in the literature, we initially investigate voice coding through interviewing seven experienced voice coders with physical impairments about their current experiences, barriers, and opportunities in their development practice. In a follow-up study, five of these participants were invited to evaluate a voice coding prototype utilising a fixed grammar approach, thus providing a mechanism to further understand their perspectives of alternative voice coding methods.

Related Work

Research studies have explored the use of voice commands to facilitate the writing, editing and navigation of syntax (León-Cordero et al., 2021; Van Brummelem, 2020; Jung et al., 2019; Memeti and Pllama 2018; Maloku and Pllana, 2016; Wagner and Gray, 2015; Delimarschi et al., 2014; Gordon and Luger, 2012; Begel and Graham, 2006; Hubbell et al., 2006; Désilets et al., 2006; Arnold et al., 2000). For instance, Rosenblatt et al. (2018) developed VocalIDE that supports voice coding through commands such as “type”, “write”, and “add” for text entry, “select” for text selection, and “change” or “replace” for altering existing text. Similarly, Paudyal et al. (2020) developed Voiceye that utilises fixed natural commands for writing, navigating, selecting, and deleting code, alongside the use of gaze and mechanical switches to support wider coding activities. Paudyal et al. (2022) also investigated how widely used navigation approaches in mainstream development environments (e.g., find, find all definitions, and find by reference) can be tailored for voice interaction. Moreover, Soto Munoz, et al. (2023) developed Mancodev that supports coding with JavaScript via fixed voice commands such as “foreach”, “while”, “camal”, and “print”.

In other work, Nowrin et al. (2022) conducted interviews with developers who have physical impairments to understand their current experiences with voice coding. Developers

highlighted challenges around speech accuracy and the time required to learn custom voice commands. Participants also identified issues such as entering variables (which often use custom names) and privacy concerns around using cloud-based speech recognisers. Outside of academic literature, there are also voice coding tools such as Talon Voice (2024) that enable developers to define and map custom vocal commands to different coding tasks (e.g. “slap” to add a semi-colon at the end of a line, “drip” to insert a comma), as well as for supporting wider control of their system (via a combination of speech and gaze control).

Whilst initial work has started to explore the perceptions of disabled developers’ experiences in using voice coding tools (Nowrin et al., 2022), there remains a significant gap in the literature exploring the coding practices of experienced voice coders, including a deeper understanding of the tools they utilise, common frustrations, and their experiences in working within mixed ability teams (i.e., collaborating with non-disabled developers). Moreover, there have been a lack of studies evaluating the use of existing voice coding approaches utilising fixed speech commands by disabled developers. Further research is therefore required to address this gap and develop our understanding of different voice coding approaches.

Study 1 – Developer Interviews

To further explore the working practices of voice coders with physical impairments, we conducted semi-structured interviews to obtain a deeper understanding of their experiences.

Participants
Seven participants with physical impairments were recruited who have experience with voice coding. Participants were aged between 20 and 41 (Mean = 29.3, SD = 7.9) with coding experience ranging between 2.5 and 27 years (Mean = 12.9, SD = 9.4). Participants work across a range of coding languages, utilising different assistive technologies to support their development practice (Table I). In terms of voice coding, six participants are utilising Talon whilst one has previous experience in using Dragon NaturallySpeaking. None of the participants reported having a speech impairment.

Apparatus
Interviews were conducted remotely on Zoom or Google Meet where participants were required to use their own computer and microphone.

Procedure
Participants were recruited via our networks and a suitable time was scheduled for the interview. A researcher initially provided an overview of the project and obtained informed consent. Participants were then asked some demographic questions, as well as questions regarding their coding experience. Participants then elaborated on any physical impairments they experience in daily life that influences their development work. We also asked whether participants work within a development team where remote collaboration is required and asked them to elaborate on their experiences. We then focused on which existing assistive technologies and platform plugins they currently use to support their development work, as well as how they utilise these tools to facilitate coding activities. Ethical approval was received from the University’s ethical review committee with all participants receiving a £25 gift voucher.

Results
Two members of the research team independently reviewed the interview recordings and captured insights from participants. These insights were then compared and reviewed to

inform the iterative development of key themes capturing participants' views. The first author then conducted a final review of the videos to ensure the derived themes detailed below reflected the points raised during the interviews.

Talon Voice Experiences: 6 participants highlighted that they use Talon as their primary tool for voice coding. P4 noted that using tools such as Talon requires some technical experience (e.g., to write Talon scripts that map voice commands to specific desired actions). Similarly, P1 and P6 highlighted that custom voice commands need configuring when working with a new programming language to ensure that voice coders can operate at a similar level to non-disabled developers. Furthermore, P4 highlighted the need to know a large vocabulary of commands which can present a learning curve for those new to voice coding. However, P1 felt that after a year of voice coding someone can be as efficient as developers using a mouse and keyboard, whilst P2 noted that they can code faster now using voice compared to when previously using traditional input devices. Another important point raised by P5 and P6 was that voice RSI (repetitive strain injury) can present a problem if users are new voice interaction. It was highlighted that this can get easier with better management of typing (e.g., through taking regular breaks), as well as through configuring one or two syllable commands to take the strain off a developer's voice. P2 commented on common accessibility issues with VS Code when using Talon where voice coders cannot easily access the sidebars and do not know what has focus within the interface. P3 highlighted that modifying code is challenging via voice tools such as Talon, as well as navigating through syntax and performing standard actions such as copy and paste.

Voice Recognition Accuracy: P4 highlighted that inaccurate voice models with "tedious" configuration options within Talon can present issues. Similarly, P2 and P5 mentioned that accidental actions triggered by misrecognition could cause frustration (e.g., when chaining multiple voice commands together). P7 noted that longer commands within Talon are typically recognised more accurately, although P1 highlighted that out-of-vocabulary words can be difficult to input using voice (these need to be entered letter-by-letter). For instance, P6 mentioned that variable names are commonly challenging to write and highlighted that accessible variable naming conventions are needed. P4 also highlighted that there is a lack of standardisation for how commands are constructed across tools, thus leading to users having to refer to cheat codes, files, or other resources.

Extension Usage: The Cursorless for VS Code extension (Cursorless, 2024) that supports manipulation of syntax via voice commands was identified by six participants as being relatively effective, although it was felt to not be intuitive due to the high number of voice commands that need to be learnt. Participants also highlighted that the extension utilises visual cues such as colour and symbols positioned around syntax which have to be observed in addition to issuing voice commands (P7 estimated that it can take around a month to become proficient with the tool). Furthermore, P7 mentioned that Cursorless cannot be used whilst editing code using the Chrome DevTools, thus presenting barriers and challenges. Four participants (P3, P4, P6, P7) also highlighted Rango (2024) as a useful browser extension for supporting web accessibility and wider development tasks (through attaching labels to all links on a web page which users can verbalise to perform a selection). P1 also mentioned that they use a Git extension for VS Code which supports certain actions via voice (e.g., "git push", "git pull", etc.).

Additional Tools: P1 commented that applications which require the use of a mouse (e.g., features based within hover states) can cause issues for voice coders. P1, P2, and P7 emphasised the need for gaze interaction to address problems associated with these types

of scenarios, while P6 mentioned that they utilise a dense mouse grid for performing clicks. Two participants mentioned that they use foot switches to facilitate stable input (P1 and P6) – for example, P1 highlighted how they use it to enable or disable eye gaze interaction. P2 felt it was important to consider mapping all actions to keyboard presses to allow users to carry out all programming tasks using a keyboard. P4 also mentioned that they use a multimodal approach comprised of a split keyboard and vertical mouse alongside the use of speech.

Voice Coding in Mixed Ability Teams: Participants identified challenges in using voice for remote pair programming in noisy environments due to potential misrecognition issues (e.g., in relation to completing university assignments (P4), working in cafes (P2)). P6 also highlighted some social challenges in that they do not want to interrupt team conversations by telling Talon to “wake up”. Similarly, P4 has used VS Code live share for collaborative coding and highlighted that if they are on Zoom with others and forget to unmute, colleagues can become confused when voice coders are using custom commands to generate code. In terms of video calling software, participants highlighted that it can be problematic when used with voice assistive devices and requires filtering of voice commands, normal speech, and environment noises (e.g., P6 commented that Teams “hogs” the microphone).

Summary: It was clear from the voice coders interviewed that they are currently utilising custom voice commands to support their coding practice, although they also noted challenges associated with this approach (e.g., a steep learning curve, volume of commands that need to be memorised, etc.). Another common approach within the literature is the use of fixed commands that utilise more intuitive and natural terms (e.g., “type i equals 0”, “select lines 24 to 28”, “comment [text”), although there has been limited work evaluating this type of approach with disabled voice coders. The subsequent sections address this gap through highlighting the development of a research prototype facilitating this method of voice coding, along with a user evaluation with five voice coders with physical impairments.

ID	Age/Gender	Physical Impairments	Coding Experience/Languages	Coding Tools
***P1**	35 (M)	RSI – chronic pain in wrists, inflammation in joints, can use hands, but cannot type for long periods (had to switch to voice coding).	20 years: Java servers; Typescript; C style languages; Python.	V: Talon Voice; Cursorless; Own custom extensions to support voice coding; Foot pedal. O: Git; Microsoft Teams; Zoom; Jitsi, Meet.
***P2**	35 (M)	RSI – nerve and neck pain (seeing a physical therapist since 2011).	27 years: Typescript; Python.	V: Talon Voice; Cursorless; Remapped keyboard to support voice coding. O: VS Code; Discord.
P3	20 (F)	Chronic pain exacerbated by fine motor movements – issues with typing.	2 ½ years: Java; Python; C++.	V: Talon Voice; Cursorless; Rango. O: Jetbrains; Visual Studio; VS Code.
P4	21 (M)	Chronic pain manifesting as carpal tunnel syndrome.	5 years: Python.	V: Talon Voice; Cursorless; Rango; Vertical mouse; Split keyboard. O: VS Code; VS Code Live Share; Zoom.

***P5**	41 (M)	RSI – pain in wrists, working under pressure/stress can result in more pain.	20 years: JavaScript; TypeScript; Node; React.	V: Tried Dragon Naturally Speaking previously, although not currently using voice for coding. O: Webstorm; Slack.
***P6**	25 (M)	RSI – can type a little with hands, but prefers to avoid.	10 years: Python; Java; JavaScript; TalonScript.	V: Talon Voice; Cursorless; Rango; Foot pedal. O: GitHub; ZenHub; VS Code.
***P7**	28 (M)	Pain on the side of wrists when typing.	6 years: HTML; CSS; JavaScript; Ruby on Rails; NodeJS; Ember, ReactJS.	V: Talon Voice; Cursorless; Rango; Keyboard mapping extensions. O: VS Code; Zoom; Eye tracker for mouse control; Vimium (chrome extension).

Table I: Participant details. Participant IDs with a "*" prefix represent participants involved in both study 1 and 2. In the "Coding Tools" column, "V" refers to tools specifically used to support with voice coding, whilst "O" captures other tools or software used to support each developers' coding practice.

Voice Coding Research Prototype

The research prototype was built upon an open-source code editor (CodeMirror, 2024) and was developed using HTML, CSS, and JavaScript. The main interface adopted a similar design to mainstream development environments such as VS Code (Figure 1) to mirror the mainstream applications that voice coders are currently utilising. We also integrated collaborative features such as a shared code editor where multiple users can log into a coding session, thus supporting research facilitators in discussing features during evaluation sessions. Furthermore, the prototype was designed to support HTML, CSS, and JavaScript syntax where HTML files could be viewed via a "Preview" button at the top of the interface. Autocompletion was also supported, so that when users verbalised characters a pop-up was displayed that showed completion options. Each of these options had a number associated with it that users could verbalise to action their desired completion. Similarly, numbers were placed next to files located in the file explorer to support users in opening files via vocally specifying the numbers associated with them (e.g., "open file 1"). This was implemented to address the potential challenges in verbalising the names of files which may in some scenarios present recognition issues (e.g., where filenames have used custom naming conventions). This approach also aligns with the functionality of plugins such as Rango which participants highlighted in the interviews as being a key tool in their development practice.

The speech recognition architecture consists of three elements: (1) a speech recognition component to convert speech into text (using the Web Speech API), (2) a command interpreter to process user commands, and (3) an execution routine that processes actions based on the commands issued by users. The recognition system can be initiated using the "alt" key which results in a microphone icon at the top of the page subtly pulsating to highlight that the system is listening for input. This could also be activated via alternative input devices (e.g., foot switches) to ensure it was fully accessible for voice developers utilising alternative input tools (as identified during participant interviews). Once a user issues a command, the recognised text is positioned next to the icon to provide feedback to users (consistent with the approach used in Talon). A fixed grammar was developed that provides users with specific commands for writing, editing, and navigating code (Table II). In this approach, voice input is initially converted to text and then compared against the defined

grammar (using a rule-based parser) – appropriate coding actions are then performed if a match is detected. The choice of commands were informed through previous work in the field utilising fixed grammar approaches (Paudyal et al., 2020; Rosenblatt et al., 2018), as well as some refinement by the research team based on informal testing sessions.

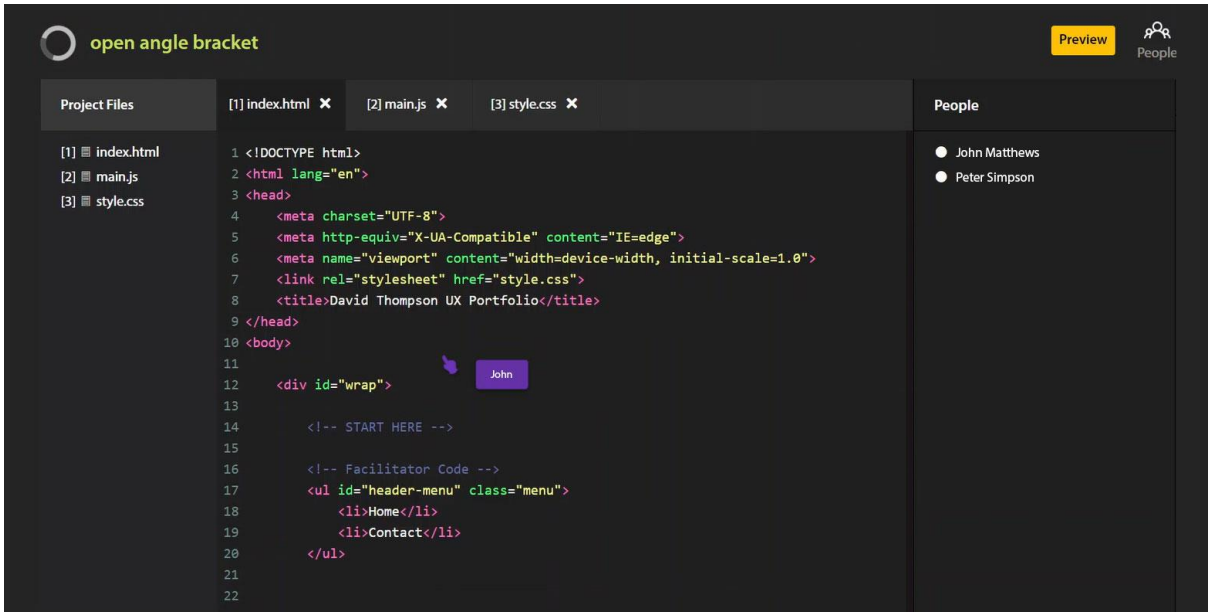

Figure 1: Screenshot of the editor interface highlighting the file explorer on the left-side, the editor in the central area, and the list of users on the right-side panel. File tabs are also displayed above the code editor where users can switch between the currently opened files. The hand with a name associated with it highlights the position of a user in the shared interface.

Voice Commands

Writing Code and Comments

“type” [“capital” / “camel” / “character”] [text/character]

“open/close” [“round bracket”, “square bracket”, “open/close curly bracket”, “open angle bracket”]

“symbol” [“and”, “around”, “asterix”, “backslash”, “cap”, “colon”, “comma”, “dollar”, “dot”, “equal”, “equal greater”, “equal less”, “hash”, “line”, “minus”, “percentage”, “plus”, “plus equals”, “minus equals”, “quotes”, “semicolon”, “single quotes”, “slash”, “underline”]

“command” [“enter”, “escape”, “space”, “tab”]

“complete” [integer, “one”-“ten”, “up”, “down”]

“comment” [text]

Navigating and Selecting Code

“scroll” [“top”, “bottom”, “up”, “down”]

“line” [integer, “up”, “down”, “start”, “end”]

“cursor” [“left” x, “right” x, “up” x, “down” x]

“select” [“left” x, “right” x, “up” x, “down” x, “lines x and y”]

“find” [text]
Editing Code
“copy/paste selection”
“undo/redo”
“delete” [integer, “line”, “line” x, “selection”]
“backspace”
File Management / Code Preview
“open/close” [“file” integer]
“preview”

Table II: List of voice commands supported by the research prototype.

Study 2 – Voice Coding Prototype Evaluation

A user evaluation of the research prototype was conducted with disabled voice coders to understand further their perceptions of common fixed grammar voice coding approaches that have been detailed within the literature.

Participants
All participants involved with the interviews in the first study were invited back for this follow-up evaluation with five agreeing to participate (denoted by a “*” character in the ID column within Table I).

Apparatus
All sessions were conducted remotely via Zoom. Participants were required to use the Google Chrome browser to ensure compatibility with the Web Speech API. Participants were also required to use their own microphones for voice input, as well as a keyboard (i.e., the tab key) or any other switch to toggle the speech recogniser.

Procedure
Ethical approval was initially received from the University’s ethical review committee. Participants joined a scheduled Zoom call where the research facilitator provided an overview of the study and obtained informed consent. They were then provided with a link to the research prototype which they opened within the Chrome browser. The research facilitator also sent a cheat sheet to participants that contained all available voice commands and then shared their screen with the participant (which contained the research prototype within the browser). The researcher gave a demo of the system and wrote some sample HTML and CSS code using the voice commands. In terms of HTML, this included an unordered list element (containing ID and class attributes) with two child elements (within a standard skeleton HTML document). For CSS, this included the styling for the unordered list that was added including the use of attributes such as “padding” and “background-color”. The participant was then asked to recreate the code underneath the syntax added by the researcher as a starting point, as well as being encouraged to write any other syntax. Whilst

working on the task, participants were asked to share any feedback regarding the voice coding approach. The System Usability Scale (SUS) was also administered after using the prototype (Bangor et al., 2009), followed by a semi-structured interview exploring perceptions of the voice coding approach. This methodology builds on other related research studies exploring voice interaction for disabled users where participants are given representative coding tasks to complete and then asked to provide feedback via a combination of post-task interviews and usability surveys (e.g., Paudyal et al., 2020). Testing sessions lasted an average of 48:05 minutes with all evaluations recorded with participants' consent for future analysis.

Results

Two researchers independently reviewed the video recordings to extract the points raised by participants during the tasks and post-task interviews. These insights were discussed by the research team and used to derive the key themes detailed below. The first author also conducted a final review of the videos to ensure that all key points had been captured. Whilst developers were able to write and edit code using the approach provided, there were a range of issues identified by participants. This was also corroborated by the SUS responses where a mean score of 51 (SD: 12.6) was captured across participants which can be classed as exhibiting an "OK" level of usability (Bangor et al., 2009).

Task Completion/Interaction Strengths

All participants were able to write code using the relevant voice commands. P5 highlighted that one advantage of this approach was "zero setup" was required to access the development environment due to the application being hosted in the cloud (i.e., no installations are required). Furthermore, P7 highlighted that the commands are appropriate for beginners who are less familiar with voice commands, whereas with Talon there is a significant learning curve where it can take around a month to become familiar with the vocabulary. P5 also commented that they could see benefits in using the prototype on the Web and that this could be useful for some people (i.e., developers who want to use the system every now and then for particular tasks).

Verbose Voice Commands

P1 mentioned that the use of two words in some commands was too verbose (e.g., "command space" to add a space) and that they would prefer monosyllabic commands. However, they noted that the downside of monosyllables is it is easy for them to overlap and can result in misrecognitions, so they need to be very distinct. P1 further highlighted that developers have to experiment with synonyms and find what works optimally for them (especially for different accents) – for instance, they mentioned that "space" did not work well for them, so they use "void" instead. P1 also discussed how they would write code using their own current approach – for example, in Talon, "angle" and "rangle" would be used for adding left and right brackets, along with the phonetic alphabet for characters (e.g., they would state "angle link ink rangle contact" in a single command to write "<li>Contact"). P2's comments were also in alignment where they felt the approach was approximately ten times slower than their normal workflow. They commented that the commands were "super verbose" - for instance, for empty angle brackets, instead of saying "open angle bracket" followed by "close angle bracket" they would simply state "diamond". Similarly, P7 commented that the commands are lengthy (e.g., "open angle bracket" is too long) and highlighted that they typically use snippets or autocompletion in normal coding to speed up the process. P6 highlighted a phonetic alphabet is needed to distinguish between letters and noted the importance of being able to customise commands.

Context Awareness

P1 mentioned that they wanted the system to have some “context awareness” when coding. For example, if the system recognises a vocal command as “command center” (which is not relevant in the current context), the system should know that the user probably meant “command enter”. Similarly, P1 highlighted that with Talon, if a user states something like “command” it knows there can only be so many options following this input (i.e., the prefix helps to narrow down the next available commands). P1 highlighted that the current system simply attempts to understand the text, but does not understand the grammar and will therefore not be as effective. Similarly, P5 also wanted the system to be context aware – for instance, if a user has just written some HTML and then navigates to a CSS file, it should be aware and presume that you may want to style those elements that have been added.

Command Chaining/Code Selection and Navigation

P1 highlighted that the chaining of commands is particularly important for expert voice coders. Similarly, P2 and P7 tried chaining commands together during the evaluation (e.g., “type menu cursor right angle”) – P2 highlighted that this is essential otherwise it would not be possible to get close to keyboard performance. P6 explicitly highlighted that they would not use a system that did not support chaining, although they highlighted that whilst Talon provides more complex command sets, it can also take longer to learn. Furthermore, P5 commented that when working on large codebases it is particularly important to be able to efficiently select and navigate syntax and that it is important to drag and drop syntax without the need for manually controlling a cursor. In particular, P6 and P7 both reiterated that they use the Cursorless extension (which supports command chaining) to facilitate these actions. P7 also mentioned that a level of control around the amount of scrolling desired is required (e.g., “scroll 10”) to provide developers with more control.

Speech Recognition Issues

P1 felt that the delay from issuing a command to the associated command being actioned was a significant issue and highlighted that the system has to be reliable and fast to support completion of tasks (the research prototype uses a cloud-based recogniser whereas Talon is installed on the client side, typically resulting in faster recognition speeds). P5 experienced some recognition issues and similarly highlighted that professionals will want a faster system that does not require calls to the cloud (which can slow developers down). Furthermore, P6 highlighted that the recognition seemed to be less accurate than Talon’s free version. P5 issued commands close together which was causing issues for the recogniser (e.g., “type s” was recognised as “type-s”). P1 also noted that Talon creates a “graph” of an individual’s commands and maps common misrecognitions to actual commands to support improved recognition accuracy.

Discussion

The research studies conducted present new insights around voice coders’ experiences in writing code. The initial interviews highlighted current working practices with common themes focused on challenges with speech recognition accuracy, the learning curve associated with learning custom commands, use of additional extensions to support code navigation and selection, and the risks associated with voice RSI. Participants also shared insights around the need for multimodal methods for coding purposes, social issues within mixed ability development teams, and the potential for voice coding performance to be comparable with non-disabled developers using a mouse and keyboard. We also gained new insights in a follow-up study evaluating fixed voice command approaches commonly

detailed in the literature. Whilst participants were able to write and edit syntax, they highlighted significant issues from their perspective in that they felt commands were too verbose (thus impacting coding efficiency), a need for context awareness and command chaining features, as well as a requirement for enhanced navigation and selection tools.

The findings indicate that future research needs to move beyond the use of fixed grammars utilising natural commands that are widely detailed within the literature (Paudyal et al., 2020; Rosenblatt et al., 2018) and instead focus on examining further how systems such as Talon (which support custom scripting approaches) can be enhanced to support developer practice. However, it is important to note that our participants were experienced voice coders who have tailored their practice to support working at a professional level. Therefore, whilst they felt the system was not suitable for professional developers, it was highlighted by P7 that this approach might be more suitable for those less experienced with voice coding, prior to moving onto more advanced configurations. It will therefore be important to explore the perceptions of the prototype with novice voice coders, as well as the transition from more verbose systems utilising natural commands to those such as Talon adopting custom scripting approaches.

Furthermore, participants also highlighted the multimodal nature of their coding practice that typically involved using voice interaction in combination with other input modalities (e.g., gaze, foot pedals, and custom keyboards). Whilst initial work has investigated multimodal voice coding approaches (Delimarschi et al., 2014; Paudyal et al., 2020), this remains an underexplored area. For instance, as noted by participants, there can be challenges when using voice as a mouse substitute in some scenarios (e.g., performing selections). In these cases, additional input modalities such as eye gaze or head tracking can provide viable alternatives, although further research is required to understand the optimal input modalities.

Future research is also required to explore the potential of large language models (e.g., ChatGPT) to support voice coders. Studies have shown that these systems can enhance productivity for non-disabled coders (GitHub, 2022), although there remains limited understanding around the impact for disabled developers. This approach might provide a more natural and flexible coding experience, although it also presents challenges in accurately interpreting a coder's intent and in producing consistent outputs. Whilst fixed grammar systems allow users to issue more precise commands that are less likely to be misinterpreted (thus leading to more consistent results), they can also present a range of challenges (e.g., learning custom commands). The combination of both methods for different voice coding scenarios may potentially provide an optimal solution, although further work is required to determine the efficacy of this approach.

In terms of limitations, it is important to note that this research was conducted with experienced voice coders, so the perspectives shared have to be considered within this context (i.e., in terms of their negative perceptions of the research prototype). The insights were also collected over a single study session, whereas it will be important in future work to conduct longitudinal studies to obtain additional insights around voice coders' working practices. Future studies should consider further analysis of voice coders development setups (e.g., types of equipment used), as well as previous voice development experience and the impact on overall speech coding outcomes.

Moreover, the inclusion of participants with a wider range of conditions and symptoms (e.g., physical impairments contributing towards cognitive fatigue) is also a crucial area for further research to understand broader voice coding considerations. Additional quantitative methods

could also be employed across a wider range of tasks to obtain further insights around user satisfaction and task performance metrics. Furthermore, the study only involved working with HTML and CSS, whereas it will be important in future work to explore voice coding experiences across different platforms and languages. However, despite these limitations, the research studies conducted present important new insights around the preferred working practices of professional voice coders that highlight research areas requiring further investigation.

Conclusion

The paper presents new insights around the working practices of developers with physical impairments who are experienced in using voice coding as an alternative development approach. An initial study with seven disabled developers identified new insights around their working practices using applications such as Talon Voice, as well as some of the challenges they experience. We also obtained their perceptions on fixed grammar coding approaches using natural language commands with findings demonstrating that experienced coders find this approach tedious, slow, and not suitable for professional level work. This research therefore presents new insights and opportunities to further investigate alternative development approaches that can support voice coders with varying levels of experience.

References

Arnold, S.C., Mark, L. and Goldthwaite, J. (2000) "Programming by voice, VocalProgramming", In Proceedings of the fourth international ACM conference on Assistive technologies, pp. 149-155.

Bangor, A., Kortum, P. and Miller, J. (2009) "Determining what individual SUS scores mean: Adding an adjective rating scale", *Journal of Usability Studies,* Vol. 4 No. 3, pp. 114-123.

Begel, A. and Graham, S.L. (2006) An assessment of a speech-based programming environment. In Visual Languages and Human-Centric Computing (VL/HCC'06), pp. 116-120).

Cursorless (2024) "*Cursorless Website"*, available at: https://www.cursorless.org/ (accessed 29 February 2024).

De León Cordero, D., Ayala, C. and Ordóñez, P. (2021). "Kavita Project: Voice Programming for People with Motor Disabilities". In Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility, pp. 1-3.

Delimarschi, D., Swartzendruber, G. and Kagdi, H. (2014) "Enabling integrated development environments with natural user interface interactions", In Proceedings of the 22nd International Conference on Program Comprehension, pp. 126-129.

Désilets, A., Fox, D.C. and Norton, S. (2006) "Voicecode: An innovative speech interface for programming-by-voice", In CHI'06 extended abstracts on Human Factors in Computing Systems, pp. 239-242.

GitHub (2022) "Research: quantifying GitHub Copilot’s impact on developer productivity and happiness", available at: https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/ (accessed: 20 June 2024).

Gordon, B.M. and Luger, G.F. (2012) "English for spoken programming", In The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems, pp. 16-20.

Hubbell, T.J., Langan, D.D. and Hain, T.F. (2006) "A voice-activated syntax-directed editor for manually disabled programmers", In Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 205-212.

Jung, H., Kim, H.J., So, S., Kim, J. and Oh, C. (2019) "TurtleTalk: An educational programming game for children with voice user interface", In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-6.

Maloku, R.S. and Pllana, B.X. (2016) "HyperCode: Voice aided programming", *IFAC-PapersOnLine*, Vol. 49 No. 29, pp. 263-268.

Memeti, S. and Pllana, S. (2018) "PAPA: A parallel programming assistant powered by IBM Watson cognitive computing technology", *Journal of Computational Science*, Vol. 26, pp.275-284.

Muñoz, J.G.S., Bringas, J.A.S., Encinas, I.D., León, M.A.C. and Verdugo, A.I.D.C. (2023) "Source code editor using voice commands to support people with motor disabilities". *Universal Access in the Information Society*, Vol. 22, pp. 1117-1134.

Nowrin, S., Ordóñez, P. and Vertanen, K. (2022). "Exploring Motor-impaired Programmers’ Use of Speech Recognition", In Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 1-4.

Paudyal, B., Creed, C., Frutos-Pascual, M. and Williams, I. (2020) "Voiceye: A multimodal inclusive development environment", In Proceedings of the 2020 ACM Designing Interactive Systems Conference, pp. 21-33.

Paudyal, B., Creed, C., Williams, I. and Frutos-Pascual, M. (2022) "Inclusive Multimodal Voice Interaction for Code Navigation", In Proceedings of the 2022 International Conference on Multimodal Interaction, pp. 509-519.

Rosenblatt, L., Carrington, P., Hara, K. and Bigham, J.P. (2018) "Vocal programming for people with upper-body motor impairments", In Proceedings of the 15th International Web for All Conference, pp. 1-10.

Talon (2024). "*Talon Voice Website",* available at: https://talonvoice.com/ (accessed 29 February 2024).

Tejada, D. (2024) "*Rango Website",* available at: https://github.com/david-tejada/rango (accessed 29 February 2024).

Van Brummelen, J., Weng, K., Lin, P. and Yeo, C. (2020) "CONVO: What does

conversational programming need?", In 2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 1-5.

Wagner, A. and Gray, J. (2015) "An empirical evaluation of a vocal user interface for programming by voice", *International Journal of Information Technologies and Systems Approach (IJITSA)*, Vol. 8 No. 2, pp.47-63.